

Privacy-enhanced Identity Management

From Cryptography to Practice



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Dieter M. Sommer

Dissertation

Privacy-enhanced Identity Management

From Cryptography to Practice



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)

von

Dipl.-Ing. Dieter M. Sommer

aus Bruck an der Mur (Österreich)

Referenten:

Prof. Dr. Michael Waidner

Prof. Dr. Stefan Katzenbeisser

Dr. Jan Camenisch

Tag der Einreichung: 18. Mai 2013

Tag der mündlichen Prüfung: 11. Juli 2013

Hochschulkennziffer: D 17

Darmstadt im Juni 2014

© 2014 Dieter M. Sommer

All rights reserved. No part of this publication may be reproduced in any form by any means without prior explicit permission by the copyright holder.

Typeset by the author in Computer Modern Roman 10 using L^AT_EX 2_ε.

Acknowledgements

First and foremost I would like to thank Jan Camenisch for making me consider a PhD thesis in the field of data privacy, for giving me guidance in my work, for always being available, when needed, for discussions related to my research, and for being my industry-side advisor for my PhD thesis. Large parts of the thesis work have been done while being in the team of Jan at IBM Research – Zurich in Switzerland. It has always been enjoyable and fruitful to work with Jan over the course of many years at IBM Research, whether in relation to this thesis or other aspects of my work, starting with my first internship at IBM Research over a decade ago.

I am very thankful to Prof. Michael Waidner for being the academic advisor of my thesis, giving me the opportunity to work as an external PhD student at his Institute for Secure Information Technology (SIT) at the Fachbereich Informatik of Technische Universität Darmstadt, and for supporting me in my thesis-related work from the very beginning. I would also like to thank Prof. Stefan Katzenbeisser of Technische Universität Darmstadt for agreeing to be the academic co-examiner of my thesis. Particular gratitude also goes to Mrs. Andrea Püchner, the personal assistant of Prof. Michael Waidner, for her continued support regarding the administrative processes related to a dissertation at Technische Universität Darmstadt, which was particularly helpful when being an external PhD student living hours away from the university. Also I would like to thank all administrative staff at TU Darmstadt I had to do with for the always friendly service provided in answering questions or handling requests related to my promotion.

Also, I would like to thank Prof. Max Mühlhäuser, Prof. Dieter Fellner, and Prof. Melanie Volkamer for being part of my PhD examination committee and for their time spent on this.

I am particularly thankful to my friends, office mates, and coworkers Patrik Bichsel and Franz-Stefan Preiss for having frequently helped out on L^AT_EX-related questions and having shared their opinions on my questions related to typesetting or phrasing sentences when time was tight in the final writing phase, for comments on my work, for the great and fun collaboration on projects and research papers, and particularly for the awesome time we have had in our office.

I would particularly like to thank Samuel Müller, a friend and former office mate at IBM, for the intense collaboration on security, privacy, and other aspects of electronic social networks and the otherwise great time during our stay at IBM in our shared office.

Furthermore, I would like to express my particular thanks to Samuel Burri and Maria Dubovitskaya, friends and former office mates at IBM, as well as Anthony Bussani and Kris Haralambiev, friends and colleagues from IBM Research, for the great time in our office, respectively for frequently trying to convince me to finally submit my thesis and stopping spending further time on it. Additionally, I would like to thank Stephan Krenn, a friend and colleague from IBM, for the few focused, yet excellent, discussions we had related to details of cryptographic mechanisms.

Especially, I am grateful to Sebastian Mödersheim, a friend, colleagues, and coworker, formerly at IBM, for the collaboration on research as well as the fruitful discussions regarding my work, particularly the formal logic aspects thereof.

Furthermore, I would like to thank all my other friends at IBM, or formerly at IBM, not mentioned already, for having made my time in the IBM Zurich Research lab and my PhD work even more enjoyable, particularly Caroline Andersson, Robert Enderlein, Cédric Favre, Lukas Kull, Abhilasha Bhargav-Spantzel, Chris Vanden Berghe, Roger Zimmermann, Toke Meyer Andersen, Olya Ohrimenko, Ashish Garg, Carl Binding, abhi shelat, Lydia Chen, Mathias Björkqvist, Hari Ramasamy, Thomas Groß, Clemens Schroedter, Annie Chen, Tadeusz Pietraszek, and all the others not mentioned here.

During my involvement in the PRIME research program, I had the opportunity to meet many great researchers and people. My gratitude goes to all those with whom I worked, mainly Giles Hogben of JRC, Thomas Rössler and Rigo Wenning of W3C, Stephen Crane, Marco Casassa Mont, Pete Bramhall, and Siani Pearson of HP Labs, Simone Fischer-Hübner and John-Sören Pettersson of Karlstad University, and Marit Hansen of ULD. Furthermore, appreciation goes to Eric Goderniaux of IBM and Gérard Lacoste of IBM for the excellent collaboration and for the great technical and other discussions and the great collaborative atmosphere they have always created.

Particularly, the discussions with Prof. Andreas Pfitzmann of TU Dresden were fruitful in progressing work in the privacy domain during the PRIME time.

During my time as project manager, when my research activities were largely suspended, I also had the opportunity to meet a great group of researchers in the PrimeLife project. For the excellent collaboration, I would like to thank all of them, particularly, Maren Raguse, Ulrich Pinsdorf, Pierangela Samarati, Michele Bezzi, Stuart Short, and Harald Zwingelberg, and those who have been collaborators already in the PRIME project and mentioned above. Particularly I would like to thank also Karin Pletka-Waldvogel and Gabriella Galli for assisting me at IBM with a plurality of the project's tasks.

Specifically, I would like to thank Karel Wouters and Ronald Leenes, friends and coworkers, with whom I worked in both the PRIME and PrimeLife projects, for the great, reliable, and dependable collaboration and being great colleagues during an extended period of time and for attending my PhD defense.

Particularly, I would like to thank Mario Verdicchio, a friend and coworker, for the fruitful discussions while collaborating in PrimeLife and for being a great colleague during his multiple research visits to IBM Research in Zurich during the PrimeLife project and beyond.

Appreciation goes to my managers Andreas Wespi, Douglas Dykeman, Phil Janson, and Michael Osborne at IBM who have supported my thesis-related work by making flexible working hours possible and giving me the time flexibility to align the thesis work with all the other project responsibilities, of which there were plenty during my time at IBM. Particularly, appreciation goes to Michael Osborne also for the fruitful and always very enjoyable collaboration on international standards related to privacy technologies within ISO/IEC.

Furthermore, I would like to thank Bernhard Jansen, a friend and colleague at IBM Research – Zurich, for the numerous and always interesting technical and other discussions we have been frequently having.

Needless to say, I appreciated the work with all my coauthors who have not yet been mentioned above. Furthermore thanks goes to all the other people whom I was working with or had to do with in other ways during my PhD time and who are not mentioned here for whatever reason.

Particularly, I would like to thank my friends Martin Schaffer, Anton Jäger, and Christian Thurnhofer for the great time whenever we met on visits back home in Austria.

Last and definitely not least I would like to thank my family for always supporting me during the time of my PhD thesis, in whatever form, and for the great time and food during my regular visits back home.

Dieter M. Sommer
Zurich, May 2013

Erklärung

Hiermit erkläre ich, diese Arbeit selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben.

Hiermit erkläre ich weiterhin, dass von mir bisher kein Promotionsversuch unternommen wurde.

Dieter M. Sommer
Zürich im Juni 2014

Publications

Conference and Workshop Publications

1. Patrik Bichsel, Jan Camenisch, and Dieter Sommer. *A calculus for privacy-friendly authentication*. In Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT'12), June 20–22, 2012, Newark, NJ, USA, ACM, 2012, pp. 157–166.
2. Jan Camenisch, Sebastian Mödersheim, and Dieter Sommer. *A formal model of Identity Mixer*. In Proceedings of the 15th International Conference on Formal Methods for Industrial Critical Systems (FMICS'10), September 20–21, 2010, Antwerp, Belgium, Springer, 2010, pp. 198–214.
3. Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. *A card requirements language enabling privacy-preserving access control*. In Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT'10), June 9–11, 2010, Pittsburgh, PA, USA, ACM, 2010, pp. 119–128.
4. Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. *Security and trust through electronic social network-based interactions*. In Proceedings of the 12th IEEE International Conference on Computational Science and Engineering (IEEE CSE'09), August 29–31, 2009, Vancouver, BC, Canada, 2009, IEEE Computer Society, 2009, pp. 1002–1007.
5. Jan Camenisch, Abhi Shelat, Dieter Sommer, and Roger Zimmermann. *Securing user inputs for the web*. In Proceedings of the 2006 Workshop on Digital Identity Management (DIM'06), November 3, 2006, Alexandria, VA, USA, 2006, ACM, 2006, pp. 330–44.
6. Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Groß, and Dieter Sommer. *User centrality: A taxonomy and open issues*. In Proceedings of the 2006 Workshop on Digital Identity Management (DIM'06), November 3, 2006, Alexandria, VA, USA, 2006, ACM, 2006, pp. 1–10.
7. Christer Andersson, Jan Camenisch, Stephen Crane, Simone Fischer-Hübner, Ronald Leenes, Siani Pearson, John Sören Pettersson, and Dieter Sommer.

- Trust in PRIME*. In Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'05), December 18–21, 2005, Athens, Greece, 2005, IEEE, pp. 552–559.
8. Jan Camenisch, Abhi Shelat, Dieter Sommer, Simone Fischer-Hübner, Marit Hansen, Henry Krasemann, Gérard Lacoste, Ronald Leenes, and Jimmy C. Tseng. *Privacy and identity management for everyone*. In Proceedings of the 2005 Workshop on Digital Identity Management (DIM'05), Fairfax, VA, USA, November 11, 2005, ACM, 2005, pp. 20–27.
 9. Jan Camenisch, Dieter Sommer, and Roger Zimmermann. *A general certification framework with applications to privacy-enhancing certificate infrastructures*. In Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006), May 22–24, 2006, Karlstad, Sweden, IFIP, Springer, 2006, pp. 25–37.
 10. Michael Backes, Jan Camenisch, and Dieter Sommer. *Anonymous yet accountable access control*. In Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES'05), Alexandria, VA, USA, November 7, 2005, ACM, 2005, pp. 40–46.

Journal Publications

1. Claudio Agostino Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. *Exploiting cryptography for privacy-enhanced access control. A result of the PRIME project*. In Journal of Computer Security vol. 18 (2010), no. 1, pp. 123–160.
2. Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Groß, and Dieter Sommer. *User centrality. A taxonomy and open issues*. In Journal of Computer Security vol. 15 (2007), no. 5, pp. 493–527.

Book and Book Chapters

1. Jan Camenisch, Ronald Leenes, and Dieter Sommer (eds.). *Digital privacy – PRIME – Privacy and identity management for Europe*. Lecture Notes in Computer Science (LNCS), vol. 6545, Springer, 2011.
2. Jan Camenisch, Markulf Kohlweiss, and Dieter Sommer. *Pseudonyms and private credentials*. In Jan Camenisch, Ronald Leenes, and Dieter Sommer (eds.), *Digital privacy – PRIME – Privacy and identity management for Europe*, Lecture Notes in Computer Science (LNCS), vol. 6545, Springer, 2011, pp. 289–308.

3. Stephen Crane, Siani Pearson, and Dieter Sommer. *Introduction: Privacy, trust, and identity management*. In Jan Camenisch, Ronald Leenes, and Dieter Sommer (eds.), *Digital privacy – PRIME – Privacy and identity management for Europe*, Lecture Notes in Computer Science (LNCS), vol. 6545, Springer, 2011, pp. 141–149.
4. Dieter Sommer. *Architecture*. In Jan Camenisch, Ronald Leenes, and Dieter Sommer (eds.), *Digital privacy – PRIME – Privacy and identity management for Europe*, Lecture Notes in Computer Science (LNCS), vol. 6545, Springer, 2011, pp. 151–288.

Patents

1. Samuel Muller and Dieter M. Sommer. *Negotiable information access in electronic social networks*. United States Patent Application Publication, No. US 2012/0297003 A1, November 22, 2012.
2. Samuel Muller and Dieter M. Sommer. *Access to electronic social networks*. United States Patent, No. US 8,234,300 B2, July 31, 2012.
3. Chris Vanden Berghe, Tadeusz J. Pietraszek, Jan L. Camenisch, and Dieter Sommer. *Privacy enhanced comparison of data sets*. United States Patent, No. US 7,974,406 B2, July 5, 2011.
4. Chris Vanden Berghe, Tadeusz J. Pietraszek, Jan L. Camenisch, and Dieter Sommer. *Privacy enhanced comparison of data sets*. United States Patent, No. US 7,974,407 B2, July 5, 2011.
5. Samuel Muller and Dieter M. Sommer. *Automated relationship management for electronic social networks*. United States Patent Application Publication, No. US 2010/0235886 A1, September 16, 2010.
6. Samuel Muller and Dieter M. Sommer. *Context-aware electronic social networking*. United States Patent Application Publication, No. US 2010/0228590 A1, September 9, 2010.
7. Carl Binding, Anthony Bussani, Jan Camenisch, and Dieter M. Sommer. *Interactive selection of identity information satisfying policy constraints*. United States Patent Application Publication, No. US 2010, 0100926 A1, April 22, 2010.
8. Anthony Bussani, Dirk Husemann, Ansgar Schmidt, and Dieter Sommer. *Method, system, and computer program product for providing e-token based access control for virtual world spaces*. United States Patent, No. US 8,132,235 B2, March 6, 2010.

9. Anthony Bussani, Jan L. Camenisch, Thomas R. Groß, Dirk Husemann, and Dieter M. Sommer. *Confidential presentations in virtual world infrastructures*. United States Patent Application Publication, No. US 2010/0058443 A1, March 4, 2010.
10. Anthony Bussani, Jan L. Camenisch, Thomas R. Groß, Dirk Husemann, Ansgar Schmidt, and Dieter Sommer. *Method, system, and computer program product for virtual world access control management*. United States Patent Application Publication, No. US 2009/0254968 A1, October 8, 2009.
11. Dieter M. Sommer and Samuel Muller. *Hybrid profile management for electronic social networks*. United States Patent Application Publication, No. US 2009/0248844 A1, October 1, 2009.
12. Jan L. Camenisch, Abhi A. Shelat, Dieter M. Sommer, and Roger D. Zimmermann. *Method and computer system for performing transactions between a client and a server*. United States Patent Application Publication, No. US 2007/0288750 A1, December 13, 2007.
13. Thomas R. Groß, Dieter M. Sommer, and Jan L. Camenisch. *Assertion message signatures*. International Patent Application, No. PCT/IB2007/051546, November 29, 2007.

Technical Reports

1. Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Thomas Heydt-Benjamin, Dieter Sommer, and Gregory Zaverucha. *Cryptographic protocols of the Identity Mixer library*. Technical Report IBM Zurich Research Report RZ3730, IBM Research – Zurich, March 19, 2009.
2. Giles Hogben and Dieter Sommer. *A meta-data and reasoning framework for open assertion and evidence exchange and query*. Technical Report RZ3674, IBM Research, Zurich Research Laboratory (joint research report with the Joint Research Centre (JRC)), October 2006.
3. Patrik Bichsel, Jan Camenisch, Franz-Stefan Preiss, and Dieter Sommer. *Dynamically-changing interface for interactive selection of information cards satisfying policy requirements*. Technical Report RZ3756, IBM Research – Zurich, 2009.

Abstract

People perform an ever increasing number of their interactions over electronic communication networks, which has induced a complex space of issues related to data privacy. The transition from manual record keeping to electronic data processing has greatly amplified privacy problems related to personal data processing due to powerful automated processing and knowledge induction capabilities. Users frequently need to reveal excessive amounts of personal data for obtaining access to services, typically being identified, while service providers and third parties extensively profile the users to commercially exploit their data as part of the so-called personal data economy.

The personal data required to be released in online interactions often exceeds the minimum that would be required for the service to be provided, and, worse, it typically identifies the user. This, together with the absence of widely deployed strong authentication systems, creates the risk of identity theft, e.g. based on user data leaked through privacy breaches, with severe consequences for both affected users and service providers. The personal data economy creates, due to the increasingly powerful automated induction of knowledge from (unstructured) data, further privacy problems, which are expected to worsen with the widespread adoption of Big Data analytics. The frequent mergers and acquisitions of service providers or data aggregators and related creation of large-scale databases further worsen the privacy issues.

In this thesis, we address user-centric privacy-enhanced identity management, with a focus on data-minimizing authentication of attribute statements about users, vouched for by third-party identity providers. Authenticated attributes allow for reducing the amount of data to be requested because they make background checks, e.g., for minimum age or creditworthiness verification based on identifying user attributes, obsolete. Data minimization benefits both users through stronger privacy and service providers through the reduction of data-breach-related risks and increased data quality and fraud prevention through the certification of attributes.

Main challenges for an open privacy-enhanced authentication system are realizing data minimization, trust delegation, user accountability, and attribute delegation—all in a single system. That is, revealing exactly the data required for a transaction, deciding on which parties to trust for certifying attributes or for providing other relevant knowledge, being able to hold non-identified users accountable for actions in case they violate agreed terms or for law enforcement, and allowing users

to delegate authority over certified attributes to parties they trust. Today's available systems or competing research proposals fail in addressing those challenges in an integrated manner. A system addressing those challenges needs to build upon complex composed cryptographic protocols to achieve the properties we require in a strong trust model, where each such protocol is a function of the data to be released. The cryptographic data release semantics needs to be expressed in a formal language which abstracts from the details of the cryptographic protocols and exposes semantics at the level of identity management concepts, the authentication messaging between parties needs to be integrated with standards, and the human user must be involved in transactions through simple yet effective user interfaces.

The presented work has started, as a foundation, with available cryptographic protocols capable of the data-minimizing release of certified attribute data while reducing the trust assumptions in third parties. Our goal of this work has been bringing those cryptographic protocols towards practice. We address the above-mentioned challenges through proposing a comprehensive system, designed around those cryptographic protocols, to make them applicable as part of an open real-world identity management system.

As one main contribution, we propose logic-based languages for representing data requests (policies), statements, protocol interface elements, and knowledge in the form of ontologies. The languages allow for automated processing, e.g., computing a response statement to a data request, in a logic calculus. The languages are used to govern system behavior, while allowing for hiding the underlying cryptographic protocol semantics.

We propose an abstract authentication model for privacy-enhanced authentication which expresses preconditions to authentication transactions and formalizes transformation rules for obtaining authenticated communication channels. The model formally specifies under which preconditions which authenticated communication channels between parties can be established using our authentication protocols. Authenticated attribute statements are expressed through the logic-based statement language.

We discuss how the cryptographic protocols to be executed for authenticating data-minimizing attribute statements are specified through formulae expressed in our logic-based languages and how multiple instances of the protocols relate to each other. We show how the cryptographic protocols realize the transformation rules of our authentication model.

A cryptographic protocol for releasing certified attribute statements is a protocol from a family of protocols and is a function of the statement to be authenticated. Because all protocols for all valid to-be-authenticated statements cannot be exhaustively specified, we propose a subsystem for compiling a protocol at runtime from the logic-based statement to be authenticated. This runtime generation results in cryptographic programs to be executed by a protocol interpreter. This has the advantages of handling the powerful semantics of our statement language through a multi-layer processing approach and allowing for aggressive performance improvements, among other things, through exploiting the instruction-level parallelism inherent to the derived programs at the cryptographic layer.

Those contributions give rise to an integrated system for data-minimizing authentication of certified attribute statements using cryptographic protocols, thereby solving the main challenges of data-minimizing authentication in an open system.

A major contribution of our work is the strong integration, on the one hand on the dimension of data minimization, trust delegation, accountability, and the further functionality crucial for an open privacy-enhanced authentication system, and on the other hand on the dimension of integration of the functionality into a single coherent system. Parts of our results have been validated with implementations and a use case prototype for an end-to-end authentication flow for a simplified system.

In addition to the abovementioned core contributions related to the authentication system, we have obtained complementary contributions in the areas of user interfaces for identity selection as part of the authentication process, a formal-model-based verification of a fragment of the cryptographic protocols, a taxonomy of user centrality in identity management, and a discussion of the various notions of trust in an open identity management system.

Orthogonal to the authentication system, we have, among other issues, addressed the problem of access control in electronic social networks. The results can substantially contribute to user privacy by enforcing user preferences and thereby solve a main problem in the social network space. We also discuss how social networks can be leveraged to bootstrap a public key infrastructure, and how to automate profile management in social networks. We put forth an approach for integrating privacy-enhanced authentication with a virtual world system and for giving presentations in public areas therein while ensuring confidentiality of the content.

With our core contributions related to the privacy-enhanced authentication system and the additional contributions around and orthogonal to it, we address an important fraction of the overall space of privacy issues.

Zusammenfassung

Ein wesentlicher Teil der Interaktionen von Menschen wird heutzutage über elektronische Kommunikationsnetzwerke abgewickelt, was zu einem komplexen Problemraum im Bereich des Datenschutzes führt. Der Übergang von manueller zu elektronischer Datenverarbeitung hat die Datenschutzprobleme, welche im Rahmen von Datenverarbeitung auftreten, durch automatische Datenverarbeitung und Extraktion von Wissen über die Benutzer massiv verschärft. Benutzer müssen in vielen Interaktionen unverhältnismäßig viele Daten herausgeben, um Zugang zu Services zu erlangen und sind üblicherweise identifiziert, wohingegen Diensteanbieter und Drittparteien extensive Profile von Benutzern anlegen und, als Teil der sogenannten Personal Data Economy, mit kommerziellem Nutzen auswerten.

Die persönlichen Daten, welche in online-Interaktionen herausgegeben werden müssen, überschreiten oftmals das Minimum an Daten, auf Basis dessen ein Service angeboten werden könnte. Weiters werden die Benutzer typischerweise identifiziert, auch wenn es für den Service nicht notwendig wäre. Zusammen mit der Tatsache, dass es keine weit verbreiteten Authentifikationssysteme für sichere Authentifikation im Internet gibt, kann das Identitätsdiebstahl, z.B. basierend auf von Hackern gestohlenen Datensätzen, begünstigen – mit schwerwiegenden Folgen für Benutzer und Diensteanbieter. Weitere Datenschutzprobleme entstehen durch die Personal Data Economy selbst, v.a. durch die immer besser werdenden Mechanismen, Wissen aus (unstrukturierten) Daten abzuleiten. Diese Problematik wird durch die weite Verbreitung solcher Mechanismen in Zukunft verschärft werden.

In dieser Dissertation adressieren wir benutzerzentrisches, datenschutzfreundliches Identitätsmanagement, wobei der Fokus auf datenminimierende Authentifikation von Attributtaussagen über Benutzer, welche von Zertifizierern geprüft sind, gelegt ist. Authentifizierte Benutzerattribute erlauben es, die Menge angefragter Daten im Rahmen eines Services zu reduzieren, weil dadurch Hintergrundchecks basierend auf der Benutzeridentität, z.B. für eine Alters- oder Kreditwürdigkeitsüberprüfung, unnötig werden. Datenminimierung hilft dabei sowohl Benutzern, deren Privatsphäre zu schützen, als auch Diensteanbietern, durch die Reduktion von erwarteten Kosten im Kontext mit der Exponierung von Kundendaten und durch erhöhte Datenqualität durch zertifizierte Attributsinformation, was u.a. auch zur Verhinderung von Betrug benutzt werden kann.

Große Herausforderungen für ein offenes datenschutzfreundliches Authentifikationssystem sind die Realisierung von Datenminimierung, Delegation von Ver-

trauensentscheidungen, Benutzer-Accountability und Delegierung von Attributen, unterstützt durch ein integriertes System, d.h., die Bereitstellung von genau den benötigten Attributinformationen durch Benutzer, die Entscheidung, welche Parteien für das Zertifizieren von Attributen oder die Bereitstellung anderen maschinenverarbeitbaren Wissens vertrauenswürdig sind, die Möglichkeit, Benutzer, welche gegen die Regeln eines Diensteanbieters verstoßen, haftbar machen zu können, auch wenn sie anonym interagieren, und den Benutzern zu erlauben, Attribute an andere Benutzer, welchen sie dafür vertrauen, zu delegieren. Heutige Systeme oder Forschungsergebnisse haben es nicht geschafft, all diese Punkte in einer integrierenden Art und Weise zu adressieren. Ein System, welches all dies adressiert, muss auf aus Teilprotokollen komponierten kryptographischen Protokollen aufbauen, um die Datenschutzigenschaften zu erreichen, wobei ein solches Protokoll eine Funktion über die Daten ist, welche herausgegeben werden sollen. Die Semantik der Protokolle bzgl. Datenherausgabe muss in einer formalen Sprache, welche von kryptographischen Details abstrahiert und Semantik auf der Ebene von Identitätsmanagement hat, ausgedrückt werden. Der Nachrichtenaustausch im Rahmen einer Authentifikation muss in ein standardisiertes Framework hierfür passen, welches in der Praxis benutzt wird. Weiters müssen einfache Benutzerinterfaces gebaut werden, um die Benutzer in den Identitätsmanagementprozess einzubeziehen.

Die Arbeit an dieser Dissertation hat mit den vorhandenen kryptographischen Protokollen zur datenminimierenden Herausgabe zertifizierter Attribute unter minimalen Vertrauensannahmen bzgl. dritter Parteien begonnen, mit dem Ziel, diese näher an die Praxis zu bringen. Wir adressieren die oben genannten Herausforderungen durch ein umfassendes System, welches um die kryptographischen Protokolle gebaut worden ist, um diese als Teil eines offenen Identitätsmanagementsystems für die Praxis nutzbar zu machen.

Als einen Hauptbeitrag definieren wir eine logikbasierte Sprache, welche Datenanfragen (Policies), Aussagen, Protokollinterfaceelemente und Wissen in der Form von Ontologien ausdrücken kann. Die Sprachen können automatisch verarbeitet werden, z.B. um eine Attributtaussage zu machen, welche eine Policy erfüllt. Formeln in den Sprachen drücken das Systemverhalten aus, wobei die technischen Details der kryptographischen Protokolle eingekapselt bleiben können.

Wir definieren ein abstraktes Authentifikationsmodell für datenschutzfreundliche Authentifikation, welches mit seinen Regeln die notwendigen Bedingungen für die Durchführbarkeit von Authentifikationstransaktionen ausdrückt, d.h., Bedingungen, um einen authentifizierten Kommunikationskanal aufbauen zu können. Authentifizierte Attributtaussagen werden in der logikbasierten Aussagensprache ausgedrückt.

Wir diskutieren, wie die kryptographischen Authentifikationsprotokolle durch Formeln in den logikbasierten Sprachen spezifiziert werden und zeigen, wie man mit Hilfe der kryptographischen Protokolle die Kanaltransaktionsregeln des Authentifikationsmodells realisieren kann.

Ein kryptographisches Protokoll zur datenminimalen Herausgabe zertifizierter Attributtaussagen ist ein Protokoll aus einer Protokollfamilie und kann als Funktion über die Aussage betrachtet werden. Weil es praktisch unmöglich ist, alle Protokolle

für alle gültigen zu authentifizierenden Attributaussagen erschöpfend anzugeben, diskutieren wir ein Subsystem zur Laufzeitgenerierung von kryptographischen Protokollen aus den Attributaussagen. Diese Laufzeitgenerierung resultiert in kryptographischen Programmen, welche von einem Interpreter ausgeführt werden können. Die Vorteile dieses Ansatzes liegen darin, dass die ausdrucksstarke Semantik der logikbasierten Sprache für Attributaussagen durch einen mehrstufigen Prozess zu einem Protokoll transformiert wird, und dadurch die Komplexität der einzelnen Stufen ausreichend niedrig gehalten werden kann, und dass spürbare Performanzsteigerungen für die Protokollausführung, z.B. durch Ausnutzen des den kryptographischen Protokollen inhärenten Instruktionsparallelismus, erzielt werden können.

Diese Resultate ergeben, zusammen betrachtet, ein integriertes System zur datenminimierenden Authentifikation zertifizierter Attributstatements mittels kryptographischer Protokolle und löst damit einige große technische Herausforderungen offener datenminimierender Authentifikationssysteme.

Ein Hauptresultat dieser Arbeit ist die enge Integration, einerseits im Sinne einer Integration der Funktionen miteinander, und andererseits Integration in ein kohärentes System. Teile dieser Arbeit sind durch Prototypenimplementierungen validiert worden, u.a. einem Prototypen eines vereinfachten Systems, welcher einen vollständigen Authentifikationsnachrichtenfluss realisiert.

Zusätzlich zu den oben genannten Hauptresultaten im Rahmen des Authentifikationssystems haben wir noch weitere Resultate komplementär dazu erarbeitet. Diese Resultate umfassen Benutzerinterfaces für die Auswahl von Attributaussagen, eine Modellierung eines Teils der kryptographischen Protokolle basierend auf formalen Modellen, eine Taxonomie für benutzerzentrisches Identitätsmanagement, und eine Diskussion der unterschiedlichen Bedeutungen des Vertrauenskonzeptes im Rahmen eines Identitätsmanagementsystems.

Orthogonal dazu haben wir Zugriffskontrolle zu Profildaten in elektronischen sozialen Netzwerken behandelt, wobei unsere Resultate dem Datenschutz von Profildaten zuträglich sind. Ein weiteres Resultat behandelt das Bootstrappen von Public-Key-Infrastrukturen mittels elektronischer sozialer Netzwerke. Weiters wurde gezeigt, wie man ein Authentifikationssystem, wie das hier behandelte, in virtuelle Welten integrieren kann und wie man Präsentationen mit vertraulichem Inhalt in solchen virtuellen Welten halten kann.

Mit unseren Hauptresultaten im Rahmen des Authentifikationssystems und den weiteren komplementären und orthogonalen Resultaten ist ein wichtiger Teil des Problembereiches Datenschutz in Kommunikationsnetzwerken adressiert worden.

Contents

1	Introduction	1
1.1	Motivation	2
1.1.1	Data Disclosure	2
1.1.2	Data Breaches	3
1.1.3	Personal Data Economy	5
1.1.4	Data Protection Regulations	7
1.2	Privacy	10
1.3	Data-minimizing Authentication	11
1.3.1	Cryptographic Protocols	11
1.4	Challenges of Data-minimizing Authentication	14
1.5	Contributions	16
1.5.1	Thesis Contributions	16
1.5.2	Complementary Contributions	24
1.5.3	Orthogonal Contributions	25
1.5.4	Summary of Contributions	29
1.6	The Need for Data-minimizing Authentication for the Future Internet	29
1.7	Thesis Outline	30
2	Concepts and Terminology	33
2.1	Basic Terminology	33
2.2	Data Minimization	35
2.3	Pseudonyms	37
2.4	Credentials	38
2.5	Partial Identity	39
2.6	Accountability	40
2.7	Unconditional and Conditional Release	41
2.8	Delegation	42
3	Authentication Model	45
3.1	Introduction	45
3.2	Related Work	46
3.3	Maurer–Schmid Channel Model	47
3.4	Extended Channel Model	49

3.4.1	Pseudonyms	49
3.4.2	Statement-based Party Annotations	51
3.4.3	Authenticating a Statement	51
3.4.4	Authentication and Confidentiality	53
3.5	Channel Transformation Algebra	57
3.5.1	Basic Transformations	58
3.5.2	Statement-based Transformations	60
3.6	Authentication vs. Authorization	71
3.7	Evidence and its Verification	73
3.8	Authentication with Credential Protocols	74
3.9	Registration and Derived Authentications	76
3.10	Future Work	79
4	Data Representation, Logic, Ontologies, and Semantics	81
4.1	Introduction	82
4.2	Related Work	84
4.3	Architectural Preliminaries	86
4.4	Data Model and Concepts	87
4.4.1	Basics	87
4.4.2	Connectives and Formulae	89
4.4.3	Functions of Parties	89
4.4.4	Identities	91
4.4.5	Conditionally Released Identities	95
4.4.6	Opaque Identities	97
4.4.7	Identifier Objects	98
4.4.8	Delegation of Attribute Authority	100
4.4.9	Transfer of Identities	101
4.4.10	Certification Metadata	102
4.4.11	Type System	107
4.4.12	Expressions	120
4.4.13	Relation Predicates and Equality	123
4.4.14	Property Predicates	128
4.4.15	Other Predicates	129
4.4.16	Features of an Identity	129
4.4.17	Extension: Operators	131
4.5	Data Statement Language	133
4.5.1	Syntax	134
4.5.2	Interoperability Aspects	142
4.6	Data Request Language	143
4.6.1	Syntax	143
4.6.2	Meaning of Disjunctions and Ontological Concepts	147
4.6.3	Examples	148
4.6.4	Third-party Requests	150
4.6.5	Authorization Policy Model	151
4.7	Repository and Portfolio	151

4.7.1	Identifier Relationships	151
4.7.2	Identity Relationships	153
4.7.3	Data Track	156
4.7.4	Profile Data	158
4.7.5	Implementation	159
4.8	Applying the Data Model	160
4.8.1	Establishing Objects and their Local Representations	160
4.8.2	Data Release	165
4.8.3	Conventional Certificates	171
4.8.4	Discussion	173
4.9	Calculus	173
4.9.1	Properties of the Logic	174
4.9.2	Deductive System	175
4.9.3	Axioms and Theories	177
4.9.4	Machine Implementation	179
4.9.5	Contradictions	180
4.10	Ontologies	182
4.10.1	Types of Ontologies	183
4.10.2	Composition of Ontologies	190
4.10.3	Trust in Ontologies	192
4.10.4	Interoperability Challenges	192
4.10.5	Distribution of Ontologies	194
4.10.6	Discussion	194
4.11	Semantics	195
4.11.1	Types and Interpretation Domains	196
4.11.2	Interpretation	197
4.11.3	Predicates	201
4.11.4	Interpretation for a System State	205
4.11.5	Extension: Operators on Attributes	209
4.11.6	Discussion	210
5	Cryptographic Protocol Integration	213
5.1	Prerequisites	214
5.1.1	Cryptographic Object Structures	215
5.1.2	Dummy Identities and Certificates	216
5.2	Establishing Identifier Relationships	217
5.2.1	Protocol for Establishing a Registration Identifier Relationship	217
5.2.2	Protocol for Establishing an Identifier Relationship	222
5.2.3	Multi-party and Public Pseudonyms	224
5.2.4	Traditional Technology	224
5.2.5	Relation to the Channel Model	225
5.3	Establishing Identity Relationships	225
5.3.1	Protocol for Establishing an Identity Relationship	225
5.3.2	Protocol Interface	231
5.3.3	Post Processing	236

5.3.4	Relation to the Channel Model	236
5.4	Release of Data	238
5.4.1	Protocol for Releasing Data	238
5.4.2	Term Derivation and Renaming	244
5.4.3	Post Processing	248
5.4.4	Architecture Aspects	248
5.4.5	Relation to the Channel Model	249
5.4.6	Construction of $\phi_{\mathbf{A}}$	250
5.4.7	Conventions for Requests	261
5.4.8	Collateral Release	262
5.4.9	Result Selection	264
5.5	Discussion	265
5.6	Realizing the Channel Calculus	266
5.6.1	Cryptographic Objects	267
5.6.2	Protocols	270
6	Run-time Protocol Generation	279
6.1	Overview	281
6.2	Low-level Identity Specification Language	282
6.2.1	Predicates	283
6.2.2	Transformation $\mathcal{M}_{\mathcal{L}_{LL}}$	283
6.3	Abstract Protocol Language	286
6.3.1	Language \mathcal{APL}	286
6.3.2	Transformation $\mathcal{M}_{\mathcal{APL}}$	287
6.3.3	Protocol Optimization	289
6.4	Cryptographic Protocol Language	290
6.4.1	Languages \mathcal{CPL} and \mathcal{CPL}^*	290
6.4.2	Cryptographic Building Blocks	291
6.4.3	Transformation $\mathcal{M}_{\mathcal{CPL}}$	292
6.4.4	Transformation $\mathcal{M}_{\mathcal{CPL}^*}$ to \mathcal{CPL}^*	292
6.4.5	Proof Protocols	293
6.4.6	Technical Realization	296
6.4.7	Composability of Cryptographic Protocols	297
6.5	Protocol Execution	298
6.6	Future Work	299
7	Implementation	301
7.1	Cryptographic Protocol Library	301
7.2	WS-Trust Request-Response Flows	303
7.3	User Interface	303
7.4	End-to-end Privacy-preserving Access Control	304
7.5	Ontology-based Reasoning	305

8	Conclusions	307
8.1	Open Problems	309
8.2	Outlook	312
	References	317

Chapter 1

Introduction

In the recent years, an increasing number of people's interactions has been carried out over electronic communication networks. This has given rise to a complex set of issues related to privacy in the domain of user data being held and processed by organizations.

Using services on the Web leads to the release of a multitude of data items, some being released explicitly by the user, many implicitly without the user noticing. Explicit release of data is mainly performed through providing uncertified attributes, e.g., in Web forms, interactions through Web interfaces, or uploading any kind of content to service providers. Implicit release happens through mechanisms such as tracking based on cookies or the collection of clickstream data, with the purpose of building behavioral profiles about the users [OEC08]. When using services on the Web through a web browser, almost all implicit release of data happens on the application layer, that is, through Web technologies such as cookies, related application technologies, e.g., Flash Cookies, or code embedded in Web sites.¹ At the communication layer, the Internet Protocol (IP) addresses are revealed to the other communication endpoint as part of the communication protocol.

The use of the Web in people's everyday interactions causes privacy problems of different kinds. A main issue is the lack of awareness of users regarding the implicit data release and the processing being performed, particularly the inference of knowledge through data mining, as well as the purposes of the data collection

¹Note that we often use the terms *Web* and *Internet* synonymously as is common practice.

and processing. Users are neither aware of the far-reachingness of the implicit data collection during their browsing activities, nor of the use of the data by the service providers and third parties. Related to this, the consent users give when using services may be seen as entirely insufficient. Another issue is the excessive collection of personal data, partly due to user-provided attributes not being authenticated. This leads to large collections of identifying personal data residing in databases of service providers, being at the risk of getting compromised, e.g., through hacking attacks.

This thesis defines, as its main contribution, a privacy-enhanced identity management system with a focus on data-minimizing authentication of certified attribute statements. That is, we propose technology for minimizing the amount of data that needs to be released in electronic interactions, with the advantages of certified data, where minimal means exactly the data required for providing the service. Relying on minimal data possibly requires a redesign of the underlying business processes. We have thereby overcome substantial challenges with respect to integrating technologies into a coherent system. We next motivate how such technology can help both users and service providers in today's complex Web ecosystem and outline the regulatory context. We give an overview of how we achieve data-minimizing authentication based on cryptographic protocols, discuss the main challenges for this in an open system, and summarize our contributions in this space as well as additional results. We also clarify that such technology is a solution only to a part of the privacy problem in the complex ecosystem of today's Internet.

1.1 Motivation

We motivate the need for data minimization through arguing how it can substantially improve the privacy of citizens and at the same time benefit service providers by reducing the expected risk-related cost by mitigating the effect of data breaches and potentially following identity theft and through the increased data quality, resulting in a reduction of the potential for e-commerce-related fraud as well. Also, we motivate how data minimization can help re-establish a balance in the light of the excessive information collection practices in today's personal data economy. We discuss privacy regulations as a potential driver for the deployment of privacy-enhancing technologies and the recent dynamics worldwide in this field as an indication that policy makers increasingly see the importance of data protection for the economy.

1.1.1 Data Disclosure

Today's established approach of service providers acquiring required personal data is collecting attribute information the users disclose. Some of those data items are explicitly released as uncertified *attribute claims* the user makes to the service provider. Most prominently, this is done through Web interfaces today, potentially with support through form-filling tools, and, once released, the data are associated

with the user in a corresponding *user account*. Users can log on to the account after its creation through an authentication mechanism, most prominently providing a username and password established during account creation. Further data needed for providing the service or for commercial exploitation of the data may be acquired by the service provider through profiling as explained later.

There are multiple privacy and trust problems associated with today's approach: Users need to release more data than minimally required for the service in case the service provider needs to authenticate certain attributes of the user through on-line third-party services, thereby requiring the *identification* of the user through *excessive release* of data not otherwise required for the service provision. This excessive data release may be seen as a privacy concern, because of not following regulatory principles and creating the possibility of data being exposed. Examples for third-party-authenticated attributes are minimum age or credit rating of a requester. Furthermore, attributes cannot be reliably authenticated in this setting as no strong mechanism for binding the requester to the attributes is available. This poses a trust problem related to attribute data and may result in *low data quality*, meaning that attributes may comprise unintended errors or wrong attribute values may be provided by users on purpose, e.g., to protect their privacy when excessive data are requested or in the context of criminal actions related to identity theft.

The problems of excessive data release and low data quality are rooted in that attributes users provide are neither endorsed by third parties trusted for this purpose, also referred to as *certifiers*, nor linked to the users in a strong way in that only legitimate holders of attributes can make those attribute claims. Many of today's online services could build on a small set of reliably certified attributes bound to their legitimate holders which those can authenticate to the service providers, instead of today's excessive set of attributes, thus following the principle of minimizing data release. Defining a system for authenticating data-minimizing attribute statements is the main contribution of this thesis.

1.1.2 Data Breaches

A *data breach*, or *privacy breach*, is the unintended exposure of personal data held by a party through an attack or, in a minority of the cases, the exposure through error. Alone the annual report of a major international investigation organization reports that 174 million records were compromised through 855 confirmed breaches in 2011 [Ver12]. For 2012, the same investigations team reports 44 million compromised records in 621 confirmed data disclosures. From 2004 to 2012, a total of 2500 confirmed data disclosures lead to 1.1 billion compromised records [Ver13]. Those reports combine data from 19 global organizations researching data breaches—it is unclear what the fraction between reported and investigated breaches is among the total number of breaches happening. Particularly, it is not ensured that breaches are always detected or, if detected, reported and reflected in the statistics.

Most breaches are caused through *hacking*, followed by using *malware* and *social engineering* attacks [Ver13]. Data breaches may expose huge sets of personal data records, including financial data, such as credit card records. Data breaches through

hacking are mainly caused by financially motivated attackers, targeting assets such as payment data or other valuable personal data at rest [Ver13]. Not only large corporations are targets of hackers, lately smaller firms are increasingly targeted due to their easier-to-circumvent security measures [FW11].

Exploitation of the data for criminal activities through what is referred to as *identity theft* may be an action following the breach [RSA10], particularly in the case of financially motivated organizations being the attackers.² For example, financial fraud or shopping online using stolen personal data and reshipping through a third party are common misuse actions [OEC08, Hed12]. Thus, data breaches through hacking attacks by financially motivated attackers may have a direct financial impact on citizens and cause them substantial additional trouble. Individuals subject to identity theft incur damage in terms of cost, time to restore reputation, reputation damage incurred from identity theft, and required efforts for re-establishing creditworthiness [OEC08] as a direct consequence of an identity theft and misuse incident.

Considering estimated damage figures, identity theft is a major problem having caused a direct financial damage of USD 49.3 billion to the affected consumers alone in the USA in 2006, with a total damage to the economy of USD 61 billion [RS09]. One reason why such substantial damage can be caused are the weak authentication requirements—often it is sufficient to know personal data attributes to conduct a transaction in the name of a party. Massive data breaches allow attackers to obtain large sets of records in a single attack, thereby providing potential of incurring substantial further damage through subsequent misuse of the data.

A major recent privacy breach within the Sony Playstation network has lead to the exposure of records of 77 million users, comprising personal data such as names, addresses, email addresses, dates of birth, and credit card data [BF11, SW11]. According to Sony, 10 million credit card records may have been exposed [Osa11] in this breach. An additional breach preceding this one has lead to the exposure of addition personal data records, making a total of around 100 million records held by Sony being exposed within a few days.

A company being subject to a data breach is subject to cost in different categories, such as forensic examination, breach notification, credit monitoring, public relations, legal defense, or penalties [Zur11]. The overall cost of the data breach for Sony is estimated to amount to around USD 1.25 billion resulting from, e.g., lost business, compensation cost, and new investments [Osa11]. The total costs of the breach are estimated to be substantially higher, including the decrease of the share price of the company [Osa11]. This shows that data breaches can have a substantial economic impact on a company and the financial ecosystem, which can be seen as a motivating argument for employing stronger privacy protection.

The fine, although considered high in terms of the applicable data protection legislation, imposed by the U.K. Information Commissioner on Sony related to the breach—with the reasoning that the security measures in place have not been sufficient, although the company would have had access to the knowledge and resources

²Attacks by activists are mostly targeted at causing disruption of operations of their victims, while espionage attacks often focus on trade secrets.

to keep its customer data secure [Flo13]—is a negligible cost item compared to the overall breach cost. As of now, companies incurring data breaches are hardly penalized severely in the United States [Dow08b], thus, regulatory instruments do currently not incentivize investments in technical and organizational measures for data protection or data-minimizing identity management. The regulation proposal for the EU’s revision of its Data Protection Directive phrasing maximum penalties for non-compliance as a fraction of the annual world-wide turnover of the company [Eur12] may be, if adopted, a driver for regulatory compliance and the implied technology investments and application of data minimization technologies. In the light of such penalty provisions in European legislation, storing less data and employing data-minimizing authentication techniques may lead to lower expected costs due to reduced risks of data breaches.

Prevention of identity theft, or misuse of stolen identity information, can be accomplished through the following approaches: Conservatively, identity theft resulting from data breaches can be countered by investing in data security and reacting accordingly in case of a breach. Proactively, employing data minimization when collecting customer data can solve the problem of data breaches already in its roots. A system for data-minimizing authentication is discussed in this thesis.

This discussion on data breaches and related identity theft can be used as an argument for data minimization—collecting and storing only the minimum amount of data needed—to reduce the expected cost of data breaches for companies. According to the Wall Street Journal, companies themselves have meanwhile realized advantages of data scarceness with respect to credit card data when data are not needed for clearly defined business purposes, and have realized that, in such case, storing data is creating additional risk without creating much additional value [Dow08a]. Such rethinking may lead to a change of today’s situation which is characterized by identifying personal data being collected by companies in excess and being insufficiently protected against exposure [RS09].

According to Gartner, the concern of data breaches is expected to have consequences in the form of privacy policy revisions within companies because the value of privacy and sensitive personal information and the risk of financial losses is increasingly recognized [Gar11, Cas11]. Related to this, data growth has been identified by Gartner as a major technical challenge for companies: companies need to invest in data archiving and retirement and protection in the light of growing amounts of data [MA10, AM10]—holding data securely incurs a large overhead.

1.1.3 Personal Data Economy

With the rise of the Internet and its mostly Web-based applications, a *personal data economy* of multiple hundred billion dollars annually has been evolving and is growing fast. This economy builds on collecting data about users and exploiting the data for purposes such as directed advertising on the Web. In exchange, users get access to services such as email, social networks, and the like free of monetary charge—the currency users pay with is personal data.

The personal data economy is closely related to *profiling*, that is, analyzing events

originating from one entity (the user) with the purpose of computing knowledge about the entity. Crucial technologies for collecting data in the scope of behavioral profiling are tracking technologies such as HTML cookies, Flash cookies, beacons, and browser fingerprinting [Cas12, Eck10, RKW12]. Interacting with a single web site may involve dozens of third parties tracking the user and building behavioral profiles. Also the Internet Protocol (IP) address of a user may be used for practical tracking of a user's interactions. In addition to Web tracking, profiling builds on data that service providers receive as direct consequence of the service provisioning model, e.g., emails, calendars, location data, or social network profiles. Profiling is done in relation to identified users holding accounts, as well as on Web sites where users do not need to have an account. Certain service providers, e.g., a leading mail and a leading social network provider, can associate extensive profiles to (weakly) identified customer accounts. As of today, it can be argued that the personal data economy has created a plethora of value-added services for users, though it can also be argued that the exploitation of personal data of users is going too far.

Profiling is about *knowledge*—and not only data—as the process inductively creates knowledge from the data obtained through profiling. This generation of knowledge which the profiled individuals do not have access to, shifts the balance of power between the profiling parties and the individuals being profiled [Hil06]. This asymmetry is seen as a problem as users cannot exert sufficient control over their data. It is expected that upcoming analytics over massive scales of data, also known as *Big Data analytics*, will increase the imbalance between profiling parties and those being profiled.

The knowledge inferred from profiles on the one hand gives service providers the ability of offering personalized services to the users, and on the other hand allows for serving personalized advertisements and further exploitation of the data. Personalizing services and also advertising can, to a certain degree, be seen as desirable for users, though, users should, in our view, themselves be able to establish a balance between the collection and exploitation of data and personalization of services and advertisements, thereby realizing *user control*.

Such user control over implicit release of personal data is possible to some extent [RKW12], particularly also easy to accomplish for end users by using tools [VD10]. Certain categories of data, e.g., clickstream data and data directly required for the service, will still be revealed to the service provider during an interaction. A notable recent initiative in the tracking domain is W3C's *Do Not Track*, which has attempted to allow users to specify a non-binding intention of an opt-out of Web tracking [Tra12]. The technical realization of the underlying policy mechanism is trivial—comprising essentially communication of a single bit of information—while the semantics is still under intense discussion. Preventing the implicit release of information at the communication layer can be achieved with technologies such as onion routing as proposed by Dingledine et al. [DMS04] or mix networks, originally proposed by Chaum [Cha81]. Applying a combination of the mentioned approaches can help re-establish a balance between the interests of users, service providers, and third-party data aggregators, thereby replacing today's rather imbalanced data economy.

Proponents of the unregulated exploitation of personal data frequently bring up the argument that exploiting personal data does not lead to privacy problems when data are anonymized. This argument is increasingly challenged by new results on the *re-identification* of anonymized records through linking them against external data sets [And09, Ohm10]. This argument is further underlined through the frequent mergers and acquisitions during the consolidation the Web industry has been subjected in recent years, and the resulting single points of data holdership and the implied integration of massive data sets.

1.1.4 Data Protection Regulations

Frequently, regulations are a main driver for a widespread adoption of technologies in a market. For privacy technologies this may apply as well, for which reason we discuss important regional and global regulations and recommendations for data protection next, with a focus on the European tradition. Notably, the first comprehensive national data protection regulation was the *Data Act 1973* in Sweden [Gre12a].

An early instrument regulating data protection is *The European Convention on Human Rights* of 1950 [Cou50] by the Council of Europe. Article 8 thereof stipulates that “Everyone has the right to respect for his private and family life, his home and his correspondence.” and also that interference by a public authority shall be limited for specific interests such as national security. Article 8 of the Convention can be seen as a foundation of data protection regulations. These provisions are generically applicable, that is, also to today’s electronic interactions.

The OECD has recognized the need for data protection as a prerequisite for transborder flows of personal data in the *OECD Council Recommendation* of 1980 [OEC80]. It has recognized that OECD members have different national laws and policies, though, “a common interest in protecting privacy and individual liberties, and in reconciling fundamental but competing values such as privacy and the free flow of information.” The goal of the non-binding recommendations is that member countries adopt them in their national regulations and thereby make them legally binding, with the aim of balancing free transborder flows of (personal) data and privacy rights of individuals through an approach harmonized within OECD members. The OECD recommendations have set forth a number of basic principles for data protection: the collection limitation principle, data quality principle, purpose specification principle, use limitation principle, security safeguards principle, openness principle, individual participation principle, and accountability principle. Those and related data protection principles are often referred to as Fair Information Practice Principles (FIPs). In the context of this thesis, the collection limitation, data quality, and purpose specification principles are of particular relevance as they relate to data-minimizing authentication.

The *Convention 108 of the Council of Europe* [Cou81] is the first and only international legally binding instrument in the field of data protection [Cou10] with regard to automatic processing of personal data. The Convention needs to be implemented in national legislation by the countries acceding to it in order to apply the

convention's principles. The principles stipulated therein are similar to those of the OECD recommendations. An addition are sanctions and remedies to be applied in case of non-compliance with the national instruments implementing the Convention. Convention 108 and its additional protocol on independent supervisory authorities and transborder flows of data are the baseline for currently 44 states in Europe and one in South America, which has acceded to the convention as the 45th and first non-European state in 2013 [Cou13]. A further trend towards internationalization can be observed in the current ambitions of the revision of Convention 108 [Con12].

Directive 95/46/EC [Eur95], the European Data Protection Directive, regulates the protection of personal data on the EU level. Being a Directive, it needs to be implemented through national regulations by the member states, e.g., the federal data protection law (Bundesdatenschutzgesetz) in Germany [Deu03] or the corresponding instrument in the U.K., to become effective. It attempts to balance the right to data protection of individuals with the free flow of data within EU member states. It defines general rules for lawfulness of personal data processing and stipulates, as part thereof, data protection principles similar to those of Convention 108. The Directive has been strongly influenced by the Convention 108 of the Council.

Directive 2002/58/EC [Eur02] has been issued as part of a wide legislative framework to regulate the electronic communications sector in the European Union. It regulates data retention of connection data, use of cookies, unsolicited email, and posting personal data in public directories. The European Data Retention Directive put forth by the EU in 2006 [The06] marked a significant exception to Europe's strong history of data protection by requiring that excessive amounts of traffic data related to electronic communication be stored by service providers, and thereby has opened the field for major discussions in the legal area.

Both the Convention 108 of the Council of Europe as well as Directive 95/46/EC of the EC are currently in the process of being revised. A reason for this is the recognition that changes in the technological landscape need to be considered in those instruments. The revision of Convention 108 [Con12] attempts, among others things, to further internationalize the convention with the goal of obtaining a global data privacy agreement [Gre12c, Gre12b]: "The call for global standards was repeatedly expressed by business and civil society communities..." [Cou10]. The update to the convention also attempts to take new information and communication technologies into consideration [Cou10]. The revision of Directive 95/46/EC has lead to a proposal of substantial changes [Eur12]. The type of instrument will change from Directive to Regulation, that is, it is valid in the members states without the need of prior implementation in national laws, with the intention of a stronger harmonization within Europe than with Directive 95/46/EC. The proposal includes far-reaching changes, such as making the so-far implicit right to be forgotten explicit, increasing penalties for severe violations to substantial levels, or to redefine the semantics of the role of data processor, to only name a few. Multiple provisions of the proposal have been under heavy critique from industry [AGBR13] and will continue to be so in the ongoing discussions and associated lobbying process of unprecedented intensity.

The above-discussed instruments reflect the European tradition of human rights

and the approach of balancing rights and interests between people, commercial institutions, and governments.

In the United States—the region of the world, where most personal data are stored and processed—no cross-sector privacy legislation for protecting consumers is in place—in contrast to the situation in Europe. There are multiple sector-specific regulations available, such as HIPPA [Con96] in the healthcare sector. Otherwise, the private sector is largely unregulated, having allowed today’s personal data economy to establish. Only recently, a need for a more comprehensive protection of the consumers has been recognized by the publication of a White House report on a *Consumer Data Privacy Bill of Rights* [The12] intended to also regulate the private sector, based on the widely accepted FIPPs principles—also comprising the data minimization, referred to as collection limitation, principle.

In the Asia-Pacific region, an overarching privacy framework has been published, referred to as APEC Privacy Framework [Asi05, Asi12]. A key goal thereof is balancing privacy protection on the one hand with cross-border information flow on the other hand, with a potential to global applicability [Asi05, Asi12]. The framework is under critique and perceived to not be a very capable privacy protection framework for multiple reasons, e.g., its weakened principles compared to the already generic OECD principles, not being compatible with the EC approach, that is, not capable of fulfilling the adequacy requirement for data protection of the EC, or having little impact on ongoing policy development in the Asia-Pacific region [Gre03, Gre04]. Notable development in the region is Singapore’s Personal Data Protection Act 2012 [Rep12] requiring, among others, consent to personal data processing.

A notable exception to data protection legislation has always been national security as already captured in Convention 108 as exception in Article 9, allowing for derogation from articles mandating data protection as far as it constitutes a necessary measure for “protecting State security, public safety, the monetary interests of the State” [Cou81]. National regulations incorporate such exceptions, resulting in different degrees of balance between surveillance and, for example, the right to privacy. Particularly some legislative acts in the USA as well as the Directive 2006/24/EC [The06] are notable in this domain. Although many privacy activists defend the opinion that some current surveillance regulations are going too far, certain measures can legitimately be argued for the national security purpose.

Contrasting the approaches to privacy regulations in the world’s regions, the European tradition balances different rights of the citizens, companies, and governments, and uses so-called omnibus legislation applying to both the private sector and the public sector, while the United States has sector-specific legislation in place e.g., HIPPA [Con96] in the healthcare sector, leading to a cluttered, less effective, legal framework lacking private sector coverage. The recent proposal of a Privacy Bill of Rights [The12] is a notable development in the United States. Asia has not progressed as far in terms of privacy protection yet, with strong regulations having come up only recently, e.g., in Singapore [Rep12], and many countries being without privacy laws, at least in either the private or the public sector or both. The Central and South American region is catching up with regulations, with multiple countries in the region having enacted privacy laws in 2012 or planning to do so in the near

term, thereby mostly following the European tradition. Australia shows a strong engagement in privacy, rather building on the European value system than that of the Asia-Pacific region. Africa is, as developing region, rather slow in the adoption of privacy regulations.

On a world-wide scale, the observation is that the strongest data protection regulations have evolved from the European value system. Another observation is that an increasing number of countries in all regions of the world enact data protection legislation [Gre12a]. Reasons are not only the protection of citizens' data, but also fostering economic development, e.g., through adequacy of data protection with the European requirements expressed by Directive 95/46/EC. Notably, provisions for data minimization and consent are prominently comprised as principles in regulations, for example, the regulations following the European tradition, as well as the proposal for a Consumer Privacy Bill of Rights in the United States. Those provisions may act as a driver for the technologies of data-minimizing authentication we propose in this thesis. Regulations usually mandate the use of *state-of-the-art* technology for fulfilling protection provisions. Through research initiatives and pilot studies, new technologies become part of the state of the art and thus closer to being mandated by regulations, or at least recommended. Having observed the cryptographic techniques we build upon from their inception, over first practical instantiations, to today's state, and considering the interest of key industry players and players in the policy debate in those technologies, we are optimistic that data-minimizing authentication technologies will be deployed within the mid-term future and that both regulators and private-sector demand will accelerate this process.

1.2 Privacy

We next give some background on what privacy means, based on earlier definitions in the literature body. The concept of privacy predates the existence of electronic communication and data processing systems. Privacy is a hard-to-define concept and, as a result, authors have proposed definitions varying in connotation and scope. Warren and Brandeis define privacy as “the right to be left alone” [WB90], a broad definition that has received substantial attention. Warren and Brandeis' publication is the first one in the USA necessitating a right to privacy [Gla79]. Westin's definition of privacy is important in the field of modern information and communication technologies. Westin defines privacy as “the claim of individuals, groups or institutions to determine for themselves, when, how, and to what extent information about themselves is communicated to others.” [Wes67] In other words, privacy is about *control* over the data related to a person through this person. This definition of privacy has influenced much of the discourse on privacy in electronic communication systems and is closely related to the concept of *informational self-determination*, as originated in Germany. We use the term privacy with the connotation of Westin's definition and stress that the possibility of data minimization implied by this definition plays a strong role in our notion.

The threat of information technology to privacy has already been observed early

on in a Supreme Court decision in 1963 in the USA, where the judge has observed that “the fantastic advances in the field of electronic communication constitute a great danger to the privacy of the individual.” [Car10] This has related to surveillance of voice telephone communication, while it is more true than ever in today’s Internet, considering the increase of processing power, degree of interconnection, decrease of storage cost, automation of processing and generation of inferred knowledge, governmental surveillance, and the size of the massively growing economy around personal data.

Overall, the privacy issues have become substantially more complex since the rise of electronic data processing and communication compared to the times of non-electronic record keeping and processing. Automated data processing allows for unprecedented ways of induction of knowledge through data mining such that today’s concept of user consent is at stake. It can be argued that today’s tracking and automated data processing practices severely undermine users’ privacy.

In the government surveillance context, an argument frequently brought up against privacy is that government surveillance is not problematic if people have nothing to hide. This “nothing to hide” argument is so frequently used so that it has been investigated and found to be flawed in its roots because of assuming the equivalence of privacy and keeping (illegal) things secret [Sol07], while privacy in fact has a much broader connotation.

1.3 Data-minimizing Authentication

The issues of excessive data release and insufficient data quality related to the disclosure of uncertified data to service providers as discussed in Sec. 1.1 can be addressed by users releasing *certified data* in an authenticated manner. Ideally, a user can authenticate exactly the attribute statement required by the service provider for providing the service to the requester, without revealing any excessive information to the transaction partner, a third-party identity provider, or further third parties. Because of the authentication property, no additional attributes need to be requested to perform a third-party-vouched authentication of the few attributes that are required authentically. Overall, this can lead to a substantial reduction in the release of attribute information for the same service to be provided. That is, the principle of data minimization can be achieved in a strong sense.

1.3.1 Cryptographic Protocols

The technology for realizing the data-minimizing authentication of certified attribute statements exists in the form of *cryptographic mechanisms*. These mechanisms comprise signature schemes and protocols with privacy-enhancing properties [CL02b, CL04], mechanisms for ensuring accountability in data-minimizing settings [CS03, CD00], as well as mechanisms for proving properties in a data-minimizing way and combining such proof protocols [Sch91, CS97a, DF02, CDS94, Cra96, Bou00, CS97b, CKY09, BKS⁺09, FS86], the security of all those mechanisms

being based on the assumption of the intractability of certain number-theoretic problems.

Additionally, when employing tamper-resistant hardware tokens with authentication mechanisms for users to those tokens and using the tokens for storing private keys and computing a subset of the cryptographic protocol steps, the binding of attributes to the authenticating user can be substantially stronger and the attributes may not be as easily exploited in the context of identity theft. Thus, it can be argued that those cryptographic protocols are the basis for a system for protecting privacy in the best-possible manner while allowing for stronger accountability properties at the same time.

The work on this thesis has started with (a subset of) those cryptographic mechanisms being available to the author and has set out to build a comprehensive system and framework around them for privacy-enhancing authentication of certified attributes. As our work inherits concepts from the more general field of privacy-enhancing identity management for privacy-enhanced authentication, we often equally use this term for referring to the concept of privacy-enhanced authentication.

The basic idea of our work is that a party (user) can obtain certificates of a special kind, denoted as *private certificates* or *credentials*. A private certificate comprises attributes about the user or a third party and comprises a signature by a certifier over the attributes. Service providers can trust certifiers for vouching for certain attributes for given purposes and phrase attribute requirements in the data request section of an authorization policy for a resource to be protected, e.g., a service they offer.

A user, when attempting to access a protected resource, can use private certificates she holds for fulfilling the corresponding data request. To this end, she constructs a data statement that fulfills the request and creates a cryptographic proof of correctness of this statement based on her private certificates. Such a proof proves knowledge (holdership) of private certificates and private keys and additionally proves properties over the attributes, e.g., an inequality over a date of birth attribute. The service provider can verify the proof and gain, based on trust it has in certifiers vouching for attributes, assurance in the correctness of the statement made by the user. Through a binding of private certificates to private keys of the user, which may be contained on tamper-resistant hardware tokens, a strong binding of attributes to users can be achieved with all the implied properties.

The system we propose can not only be used for authenticating data-minimal attribute statements, it can also replace authentication to an established account, thereby replacing today's password-based authentication. To this end, a newly generated public key (pseudonym) can be used for each account, while all those public keys can be bound to the same private key and hardware token and thus party.

A comparable solution using traditional signature schemes such as the RSA, DSA, or ECDSA schemes [RSA78, RSA83, Nat09] for revealing attribute information could solve the authenticity problem of attributes, though, would lead to excessive release of data. This would be an entirely unacceptable situation from a

privacy point of view as the revealed attributes would carry high assurance properties, thus even worsening today's situation of excessive release where users may still provide falsified information by creating different personas for different contexts or transactions in the course of their identity management. When applying such traditional technology for re-authentication to established accounts, one would need to use a different private key for each public key and thereby run into a problem when using hardware tokens for containing the key material. The approach of a party using the same unique public key for each account it has would pose a severe privacy problem through linkability of all transactions. Following this argumentation, the traditional signature technologies are not a suitable choice for authentication in open systems such as the Internet, because they violate basic privacy principles, despite being secure. Similarly, today's identity federation protocols with online certifiers all have the property that the certifier (identity provider) learns about the transactions a user performs and can create detailed profiles of the user's interactions. Thus it becomes a single point of failure in terms of data breach or misuse. That is, all prominent identity management technologies of today for transferring certified attributes reveal excessive information—conventional certificates as in electronic identity cards based on RSA, DSA, or ECDSA signatures and protocols with on-line certifiers issuing identity statements on behalf of the authenticating parties alike.

The system and framework proposed in this thesis is capable of solving the problem of privacy-enhancing authentication, while conventional technologies would result in unacceptable privacy problems when being employed for securely authenticating attribute statements or authenticating to accounts. Using strong authentication technologies provided by our work also in the back-end systems may raise the barrier for hackers and thus help avoid data breaches, especially when considering that around 80% of all hacking attacks leading to data breaches exploit weaknesses of single-factor password-based authentication [Ver13].

Thus, our work can lead to a stronger protection of user privacy compared to today's situation both in terms of revealing substantially less data as in today's interactions and thereby mitigating the issue of data breaches by companies not storing the resulting large amounts of personal data and also mitigating further privacy issues related to excessive attribute release, as well as reducing the chance of attackers attempting to steal personal data once released and decreasing their utility. In terms of security, it can reduce the probability of user account credentials being obtained by attackers and misused when using tamper-resistant hardware tokens to contain private key material. Service providers obtain higher assurance in terms of the person authenticating an attribute statement being the person to which the statement applies, as well as the assurance that only account holders can authenticate to already-established accounts.

1.4 Challenges of Data-minimizing Authentication

A privacy-enhancing, user-centric, open identity management system to be used in a practical setting poses a plurality of technical and non-technical challenges as presented next.

First and foremost, *security* in the sense of integrity of authenticated attribute statements is a required property of an authentication system. That is, a statement that a requester authenticates through cryptographic protocols must be verifiable by its recipient with respect to well-defined semantics such that, in case the statement can be verified successfully, its verifier can be assured about its integrity. The cryptographic mechanisms referred to above in Sec. 1.3.1 can be used to realize this while maintaining privacy and user accountability at the same time. Authenticated statements can not only be statements about requesters' attributes, they can also be statements for logging on to accounts analogous to using username/password tuples today—only secure and based on a private key strongly bound to the legitimate key holder. Thus, such an authentication system will also be capable of replacing the less secure password scheme in use today. As password insecurity is not only a problem for accounts of users with service providers, our authentication system can also contribute to better security in this domain, especially as studies have shown that a majority of hacking attacks on organizations exploits weak (password-based) authentication, leading to substantial damages [Ver13].

The key capability of an open authentication system is the authentication of attribute statements. A main problem in such system is *trust management* for determining which certifiers to trust for vouching for a particular attribute and how to specify this in a data request. This is a non-trivial problem in an open world-wide system with a large number of certifiers. This challenge has been phrased in the context of trust management systems as delegation of attribute authority [BFL96, LGF03], though, without account to privacy.

A crucial feature of an authentication system of the kind we are interested in is that it protects user *privacy* by employing *data minimization*, the importance of which has been motivated extensively earlier. Data minimization in the authentication context means that requesters authenticate data-minimal attribute statements by revealing the statements together with proofs showing their correctness.

Using data-minimizing authentication compared to today's approach of excessive data collection implies the need for a *re-design of business processes* to work on the scarce amounts of data provided. In certain cases, the overall system requirements for service providers will be simplified through this, an example being an online wine shop requiring age verification for its customers. Instead of performing an age verification through a third-party service provider based on the requester's identifying attributes, resulting in a medium-assurance age attribute only and a weak binding to the requester, the attribute is authenticated by the requester itself. The redesign of business processes is out of scope of this work—we provide the technical solution.

A practical open authentication system should support *delegation of authority* over authenticating attribute statements. This is crucial for widespread use of such system, e.g., for electronic identity cards used for government interactions: The age

structure of a population is likely to imply that a fraction of the population will consist of digital non-natives who have not acquired the skills for using tomorrow's data processing systems. Delegating authority to authenticate attribute statements about them to other parties, e.g., their children, enables required government interactions for them to be conductible using the standard electronic processes. Prior work has considered delegation based on parties with a system-wide identifier, see, e.g., more recent work on trust management [BGF06, BFG10]. Such delegation can be realized as a special case of an authorization system based on our authentication approach.

Data minimization may frequently lead to anonymous or pseudonymous transactions, which may pose the challenge of maintaining the legitimate interests of the service providers and the law enforcement system. Having unconditionally anonymous or pseudonymous transactions is, in our view, not a situation desired from a societal perspective as it may put service providers and law enforcement to a disadvantage against (organized) dishonest users. For this reason, we advocate that a future authentication system needs to strike a balance between data minimization and *accountability*. With the technical approach we propose, privacy and accountability properties can be obtained at the same time, thus resulting in a balance when also considering trust model limitations, that is, additional trust required in third parties. Our system allows for privacy and anonymity particularly also in delegation scenarios in a flexible manner, such that either of the delegater or delegatee in an interaction can be held accountable, depending on the policy employed for the transaction.

An authentication system with the discussed properties needs to, in order to be practical, *integrate with standards* that are crucial for the area. Not much research work has been done in terms of standard integration that would predate our work.

A technical system for realizing the above-discussed properties needs to *integrate* those properties into a single system and various of its constituents, e.g., formal languages, cryptographic components, or the authorization system. Desparate solutions realizing only parts of the properties or such that do not consider the full technology stack may be interesting in theory, though, are of less use in practice because a large part of the challenge lies in the integration.

Major properties for an authentication system to be successful are *convenience* and *usability*. The system should make interactions more streamlined than they are today and be easy to use at the same time. For example, the password management and field auto-filling of modern Web browsers today is an example for a convenience feature. An open authentication system we describe should lead to convenient interactions by automation of routine tasks and ease of use.

The abovementioned challenges require, as basic technology, cryptographic credential protocols which are complex and hard to understand. A particular resulting challenge is the *integration* of those protocols and their authentication semantics into a coherent system.

1.5 Contributions

Our contributions related to this thesis are presented next, partitioned into the following classes: Contributions reported in or directly related to the thesis (Sec. 1.5.1), contributions that complement the further (Sec. 1.5.2), and contributions that are orthogonal to the further two, for protecting privacy from a wider point of view (Sec. 1.5.3).

1.5.1 Thesis Contributions

The contributions of this thesis are the definition of an open system and framework for user-centric privacy-preserving authentication based on cryptographic protocols, with the ability of balancing data minimization and accountability and delegation support. This, on the one hand, comprises basic mechanisms, and, on the other hand, integrates those mechanisms into a comprehensive system and framework. The contributions range from a high-level model for authentication, over a logic-based data representation and request language and a related logic-based calculus, to cryptographic protocols for implementing the semantics of statements in the data representation language using cryptographic mechanisms. Overall, this has led to interdisciplinary work between the disciplines logic, access control, and cryptographic protocols. The resulting system and framework provides solutions to the beforementioned challenges related to an open, privacy-enhancing authentication system.

From a technology perspective, our work combines concepts from both trust management and trust negotiation, while taking a privacy-enhancing approach and relating the level of logic-based languages with the underlying cryptographic protocols. Also, we support powerful delegation of attribute authority, accountability features, and mechanisms for ensuring a strong binding of attributes with parties without constraining system openness with unrealistically strong assumptions.

The thesis builds on published results of the author and extends and integrates those results towards a comprehensive identity management system. Besides the individual results, the integration is a further major contribution, particularly relevant for understanding challenges which cannot be anticipated in the limited scope of the individual mechanisms research, though, which arise only at the system perspective. Next, we present an overview of the contributions we have made in the context of this thesis and relate those contributions to the publications of the author and the chapters of the thesis.

Parts of the results have been obtained during the PRIME project [PRI08], which was also helpful in obtaining requirements for the later work on the system to obtain the generalized and extended results presented here.

Authentication model. We have extended the secure channel model of Maurer and Schmid [MS94, MS96] to obtain an abstract model capable of expressing the privacy-preserving and accountable authentication mechanisms of this thesis at

a high level. The model is a semi-formal model that is capable of expressing authentication and confidentiality properties of communication channels between parties using a *graphical channel notation* with bullet annotation symbols for channel endpoints as main syntactic element for representing security properties of channels. Authenticated properties are represented through an annotation specifying the (authenticated) data statement being made. Channel derivation rules—loosely speaking, a *channel composition algebra*—have been defined for deriving new channels with certain properties from a set of existing channels. We relate our cryptographic protocols to this model and explain which protocols are required for realizing a given channel composition rule. This model is presented in Chapter 3 and is a refinement of the formerly published model of Bichsel, Camenisch, and Sommer [BCS12a, BCS12b]. The presented model establishes a more explicit link to our data representation language without the simplifying abstractions of the original publication.

Logic-based data representation, reasoning, and ontologies. A major contribution of the thesis is a formalism based on first-order logic for representing identity data and requests thereof, alongside with the required conceptual groundwork. The result is a *data model* which is suitable for expressing certified identity statements to be released to other parties as well as other identity data required in a comprehensive identity management architecture. It serves also as the basis for a data request language and access control language based on it. The data request language is the dual language to the data representation language. A semantics for the data representation language is presented as well as the language formalization. Calculi based on extensions of multi-sorted first-order logic are discussed and we show how ontologies can be used for representing knowledge and how to perform automated deductions over formulae of the language.

Earlier work with contributions by the author on an authorization language with privacy features for exploiting (parts of) the semantics of cryptographic credential protocols [ACK⁺10] is an extension of work by Bonatti and Samarati [BS02]. The languages related to data representation and request have been presented in earlier work by the author in the context of a comprehensive identity management architecture [Som11] and are based on work on languages for privacy-preserving identity management [BCS05, CSZ06a] and an approach to ontology-based reasoning for privacy-enhanced authorization [HS06] by the author. The approach to data modeling and underlying concepts, the data request language, the logic calculus, ontology aspects, and the language semantics are discussed in Chapter 4. Based on the data representation language, a privacy-preserving authorization language and system have been defined by the author [Som11] based on the data request language presented in generalized form in this thesis.

Protocol integration. In Chapter 5, we present the integration of the protocols required for realizing our authentication system, with a focus on protocol interfaces expressed using our approach to data representation. The protocol interfaces establish a link between the authentication model of Chapter 3, the logic-based data

representation of Chapter 4, and the cryptographic protocols, the run-time generation framework for which is summarized in Chapter 6. They also relate different instances of the cryptographic protocols with each other, based on the data representation language. A much simplified protocol interface for an earlier version of the protocols [Som07] has been presented as a part of the work by Camenisch et al. [CSZ06a].

Run-time protocol generation. Another major contribution of this thesis is a framework and system for the *run-time generation* of cryptographic protocols for data-minimizing release of identity information to parties on the network. The cryptographic protocols allow for authenticating data-minimizing statements towards other parties and performing other relevant actions, based on cryptographic primitives and zero-knowledge proof of knowledge composition frameworks [CKY09, BKS⁺09]. The approach is characterized by a multi-layered processing stack for transforming a logic-based high-level data statement into a cryptographic protocol to be executed proving the statement correct. The approach of run-time generation is not only suitable for the Camenisch–Lysyanskaya signature schemes we build upon [CL02b, CL04], that is, multi-show unlinkable signatures, but also for signature schemes of the kind Brands has proposed [Bra00], that is, single-show unlinkable signatures.

The contributions in this space are multifold and comprise the definition of formal languages at different layers of the protocol generation stack, mappings between those languages, and finally efficiency improvements of the protocols. Efficiency improvements of the cryptographic protocols are possible at multiple layers of protocol generation and execution, starting with optimizations on protocols in their abstract cryptographic form, precomputation of protocol fragments, as well as exploitation of instruction-level parallelism at protocol execution time.

Earlier work of the author is related to a multi-layered stack for generating cryptographic protocols from high-level statements [Som07, BCS05]. We discuss a logic-based data representation from which to generate the protocols and sketch the protocol primitives [BCS05], while we realize an implementation of a simplified fragment of the protocols in a JAVA-based software library [Som07]. The subsequent version of the protocols is the official Identity Mixer (idemix) protocol specification of IBM [BBC⁺09, Sec10], which has been evolving. The earlier implementation [Som07] is based on a simplified 2-layer architecture.

Another contribution resulting from run-time protocol generation is our approach for specifying complex cryptographic protocols by specifying the protocols through specifying a series of compilers translating from a logic-based data representation language to a formal language specifying executable cryptographic protocols. Due to the genericity of the supported protocols, the presentation of the specification is more complex than the protocol specification of the basic protocols [BBC⁺09, Sec10] for Identity Mixer.

The work on run-time protocol generation in this thesis builds on those earlier results and takes them further, resulting in a 5-layer architecture capable of handling the complexity of the extended protocols of this thesis and ensuring their efficient

execution. Our approach to run-time generation of protocols is briefly summarized in Chapter 6—the full version thereof could not be presented due to page limitations for the thesis.

1.5.1.1 Supported Functionality

The identity management system we define has expressive capabilities that go far beyond that of related work in this domain, particularly in terms of data minimization, trust management, and accountability. Those capabilities are supported throughout the stack, ranging from the authentication model, via the data request and representation languages and computations over sentences expressed in them, to the generation and execution of the cryptographic protocols for proving, that is, authenticating, data statements to other parties.

Monotonous Boolean formulae. Our data model allows for specifying data-minimizing requests and statements. Formulae are monotonous boolean formulae, the predicates of which can make assertions about attributes of a user and machine. Data minimization is supported through the following crucial properties: (1) Predicates expressed over attributes instead of always revealing them; and (2) disjunctions over sub-formulae can further reduce the amount of information revealed about a party. Expressing disjunctions over sub-formulae takes data minimization further than only supporting predicates over attributes and goes beyond what most current approaches can support. Business processes can, in the light of data-minimizing transactions, still get the information they need to operate, while users can reveal less data than in traditional data-minimizing systems. Disjunctions are supported in the data representation and request languages, and thus the access control language, as well as on all layers of the framework for run-time generating the credential-based cryptographic protocols. Supporting disjunctions in data statements has raised challenges in multiple places of the framework and system, e.g., logic-based reasoning support over ontologies. The full support of disjunctions, and thus monotonous formulae for making identity statements is a major contribution of our work.

Delegation of authority. Overall, *delegation of authority* over identity information is an important aspect that is discussed in various parts of the work due to its importance in practical identity management systems.³ Particularly, it is handled in conceptual discussions, in the data model, its semantics, the access control policies, and the protocol generation framework, which are the main areas where this concept needs to be covered in order for it to be tightly integrated into the framework. The tight integration of delegation into relevant parts of the architecture allows for expressing policies including delegation, finding suitable pseudonyms and credentials that fulfill such policies, and using those in cryptographic protocols.

³This is not to be confused with delegation of attribute authority as discussed in trust management systems [LGF03], which we clearly also support as a key feature.

Delegation is still a missing feature in recent notable work on privacy-enhanced authorization languages, e.g., [CMN⁺10] and associated cryptographic protocols for private certificates and their implementations.

Outsourcing of trust decisions. An open identity system as the one defined in this thesis requires its players to make certain trust decisions when interacting with each other. For example, a service provider needs to decide which certifiers to trust for certifying user attributes used for a given purpose, or both a user and service provider need to decide which trustee they trust related to accountability of transactions. Such and analogous decisions cannot realistically be made by end users and in many cases they will be too cumbersome, complex, or time-consuming to be made by organizations such as service providers on the Internet. The reason for this is that we target an open system with a plurality of parties of each of the kinds, e.g., certifiers, where the number of such parties is too big that users or service providers can reasonably be assumed to have an overview over those parties and their properties. At a first glance, the intention to create an open system and parties being able to exploit this seems to create a dilemma. This has first been addressed in the context of the field of trust management [LGF00, LGF03].

We provide a solution to this dilemma by defining mechanisms for the outsourcing of trust to third parties as a logical consequence. This means that only few high-level or aggregate trust decisions need to be made by a party on which parties to trust for making trust decisions on behalf of the party. This is realized through ontologies, where an ontology is a formal specification of parts of the considered world, e.g., describing attributes or properties of parties or the relations between parties. Using those ontologies in an automated deduction system exploits the contained formalized knowledge for automatically making certain trust decisions.

Pseudonymous certifiers. As a special privacy-enhancing feature, our languages, protocols, and overall system allow for supporting *pseudonymous certifiers*, which extends the traditional and predominant model of professional identity providers. Thereby, we allow certifiers to act under a pseudonym or under specified properties (attributes) that are endorsed by certifiers as usual. The trust decisions by relying parties are then made based on such pseudonyms or certified properties of pseudonymous or anonymous certifiers. This leads to a much more open trust model where certifiers can be addressed in a general way through their properties and not only identifying attributes. This carries the idea of users authenticating attributes instead of identifying information towards certifiers. We think that on-line communities might be an interesting field of application where pseudonymous certifiers can issue identities to community users while building up reputation w.r.t. their pseudonyms.

Privacy with accountability. Defining technical means for balancing privacy and accountability is a contribution which is crucial for practical systems. Based on our early results of expressing anonymity-preserving data statements while ensuring

user accountability of the transactions [BCS05], we have integrated those results into our data model and protocols. Our system allows for transactions to be anonymous, while, under pre-agreed conditions, further (identifying) attribute information can be learnt by the relying, or a third, party, or both.

Registration. Our framework and system furthermore supports registration of parties in the system and the binding of credentials to registration credentials. A registration credential is one which is considered to offer sufficient security to prevent its illegitimate use by its holder or other parties, e.g., through sharing or theft, as well as sufficiently strong identity verification of the credential holder. An example for a suitable registration credential is a smart-card-based electronic identity document.

Certifiers have the freedom of choosing which credentials they accept as registration credentials and to issue their credentials with respect to those. Leaving such trust decisions to certifiers is, in our view, crucial for an open system. A registration domain comprises all credentials issued on a registration credential. Challenges arise in terms of authoring authorization policies related to registration credentials, particularly in the context of using ontology-based generalizations for referring to credential types or certifiers. We provide solutions to those challenges and support registration binding throughout the framework.

Bridging registration domains. Another contribution relevant in the light of making the identity management system more practical is the support for *bridging* registration domains. Bridging registration domains means that a user can make use of credentials of different registration domains in a single protocol for releasing data. We have designed our system from the logic-based data representation language to the cryptographic protocols layer to support that a party can have and use multiple registration credentials, even in a single data release protocol. This helps keep our system open by not mandating a single registration per user, which would lead to rather unrealistic assumptions.

1.5.1.2 System Aspects

Related to building a functional integrated system, we have made multiple contributions as outlined next.

Integration with standards. For being compliant with standard means of communication between parties, data requests and data statements with accompanying proof protocols have been integrated into the Web Services stack [CGS06, GSC07]. This work presents a general method for transferring the semantics of zero-knowledge proof protocols to XML signatures [ERS02]. The authors then show how this can be used to transfer the semantics of zero-knowledge proofs also to WS-Security [WSS04] and thus also WS-Trust [KN03a]. This enables a new WS-Federation Active Requestor profile [KN03b], using private certificates instead of traditional tokens, thereby immediately leveraging the advantages of private certificate systems.

A complete implementation of the cryptographic protocols [Som07] and XML messaging has been done to achieve the integration with the WS-Trust standard [CGS06]. It has been presented as part of a larger showcase [BSBC⁺07] using an older version of the cryptographic credential protocols. The mechanism can be equally applied to the extended data representation language and cryptographic protocols presented in this thesis. The work on WS-Trust integration [CGS06] is a practical example of how the limitations of current standards can be overcome for integrating novel mechanisms into standards and overcome corresponding lack of extensibility in their design.

Integration with commercial products. The implementation results have additionally been integrated with the access control and identity federation products of a major software vendor to showcase the practical feasibility of our approach.

In its entirety, the resulting prototype shows the realization of a privacy-enhanced identity management setting, including an integration with major standards as well as major commercial products in the identity and access management space. Through this prototype we could prove the technical feasibility of privacy-enhanced identity management systems in a real-world environment constrained through existing standards and products. Those encouraging results can act as a driver for the further development of credential systems and their deployment in real-world environments.

Implementation of a use case. Using the protocol implementation [Som07] and its WS-Trust integration [CGS06], an application prototype featuring a privacy-enabled healthcare scenario has been implemented and shown at the IIW 2007 event [BSBC⁺07]. The showcase was built upon the Tomcat⁴ application server, the standard access control system of which was exploited to accept authentications using credential-based proofs of attribute statements instead of username/password authentication.

1.5.1.3 An Integrated Approach

The work performed in relation to this thesis has progressed the state of the art in the area of privacy-preserving identity management based on credential systems at large as outlined above, and in orthogonal areas in various ways. The main result of the work is the definition of a system and framework for realizing protocols for data-minimizing authentication based on private credentials. Our contributions thereby are multifold and comprise contributions in multiple dimensions.

Regarding the dimension of *technical contributions*, we have obtained results related to a framework and system for privacy-preserving identity management as discussed already above.

Substantial value of our work, in addition to mentioned individual contributions, can be found in the dimension of *system integration*. Our technical contributions

⁴<http://tomcat.apache.org/>

have been tightly integrated to form a system and framework. This has required the alignment of the mechanisms with each other, that is, the technical contributions have not been considered from a limited perspective, rather, from the perspective of a holistic framework and system, with the additional challenges resulting from this.

Further value of our work results from a *function integration* on multiple dimensions: We have designed our system to exceed existing approaches in terms of privacy-related functionality. Also, it has been crucial for our work that the resulting system be an open system and that realistic assumptions for practical deployments be made. Bringing those aspects together has further increased the complexity of the work. As a concrete example for privacy functionality, we allow for data statements comprising disjunctions for further minimizing release of non-required data throughout the whole system, which has resulted in a substantial increase of the technical complexity in multiple areas. An example for functionality for facilitating system openness is the support of ontologies for, among other things, the outsourcing of trust decisions. Examples for realistic assumptions are the generic support of binding credentials to any of a set of (hardware-based) registration credentials or the support of multiple registrations of a party in the system. Combining multiple functionalities of those kinds has lead to a further increase of the complexity of the resulting mechanisms and framework and system.

Other work often neglects the system integration perspective, function integration perspective, or both. They address isolated aspects of contributions only, while neglecting the crucial integration aspects. Our holistic view distinguishes our work from much of the other work in the area and has been an overall focus. For example, many other works focusing on a system for privacy-enhanced authorization do not consider other protocols than revealing data, thus neglecting additional complications. As another example, various works address privacy-friendly credential-based access control, while neglecting trust outsourcing or specificities of registration of parties in the system. Those results thereby neglect requirements for practical deployments and are thus, in practice, not suitable as generic building blocks for an open identity management system we propose.

The approach of the tight integration of our contributions implies that not all chapters of this thesis are self-standing results, rather, some of the chapters bring together results from other chapters to obtain the integrated framework and system. An example for this is Chapter 5 which brings together the secure channel model, the logic-based data representation, and the cryptographic protocols. Particularly, this means that those results cannot be independently published because their value comes from the integrated perspective in the context of our overall work.

The complexity of some of our results, or the presentation thereof, has resulted from their tight integration. Many of the aspects can be much simplified when considering isolated results, e.g., the data request language without the aspects of how it integrates in terms of the cryptographic protocols.

Building a system based on the concepts we have discussed is a comprehensive effort. What has been presented earlier as contributions of this thesis forms the conceptual and technical core of such system. Additionally to this, further elements need to be thought of in a complete implementation, which have only been touched

on in this thesis. This comprises the support of standard Public Key Infrastructure (PKI) mechanisms for distributing public keys, ontologies, and the like in an integrity-protected way. This requires extensions, also of the underlying standards, for handling items not being public keys of traditional cryptographic schemes for which PKI standards exist and have originally been defined for.

An integrated system for user-centric privacy-enhanced identity management with a focus on data minimization co-designed by the author has been proposed by results of the PRIME project [CSS⁺05, CLS11]. A system prototype of it has been built as part of the project [PRI08, CLS11]. Within this project, a subsystem has been built for privacy-enhancing authorization based on credential protocols as one part of the project efforts. Ontology-based reasoning support was demonstrated to be feasible as well. Further work done in the project includes the (semi-)automated assurance assessment of parties included in the negotiation flow, life-cycle data management, usability, or anonymous communication.

1.5.2 Complementary Contributions

We next present further results related to the framework and system presented in this thesis. Those results complement the presented results by being closely related to or extending the work.

Taxonomy. User-centricity is a significant concept of federated identity management as it relates to strong user control and privacy. As multiple different notations of user centricity have been used in the identity management community, its meaning has been unclear. The author considers user centricity abstractly and establishes a taxonomy to clarify its meaning [BSCGS06, BSCGS07]. These works show the different mechanisms to realize user-centric systems and the different resulting trust assumptions. The authors classify the mechanisms into two predominant variants of user-centric federated identity management systems: credential-focused systems with offline identity providers and long-term credentials at a user's client, and relationship-focused systems with online identity providers that issue short-term credentials during transactions.

Formal model of Identity Mixer. A complementary result to the work presented in this thesis is a formal model of the Identity Mixer credential protocols [CMS10]. The model has been successfully checked using Mödersheim and Viganò's OFMC model checker [MV09a]. This result nicely complements the framework and system presented in this thesis in increasing the assurance we can have in the underlying cryptographic protocols through model checking of a simplified variant thereof.

User interfaces. We have obtained results in the area of user interfaces for choosing identity information to release in order to fulfill a data request (policy) of a requesting party, henceforth also referred to as *identity selection*. In Camenisch et

al. [CSSZ06], we have shown an approach for identity selection tightly integrated with the Firefox Web browser and consuming minimal browser area, while being easy to use. This has been fully implemented together with the credential system protocols and part of our healthcare prototype of Bhargav-Spantzel, Bussani et al. [BSBC⁺07]. In our later work, we have investigated a user interface for identity selection which allows for greater expressiveness of the underlying specification language [BCPS09]. The resulting interface consumes more screen space, though can cope with the higher expressiveness, while providing for more explicit consideration of informed consent requirements as put forth in the European Data Protection Directive [Eur95]. Another approach for the dynamic selection of identity information has been attempted [BBCS10]. All those results on user interfaces are based on simplified data representation languages compared to the one presented in this thesis. We leave the challenges resulting from a user interface capturing the complexity of the language of this thesis to future work and think that preferences-based decision support is required to handle the resulting complexity. Our results precede relevant conceptual work on user interfaces for privacy-enhancing authentication and privacy management in general resulting from the PrimeLife project [WAFH11, AFHWP12] and are among the earliest results in this area.

Notions of trust. The term *trust* has a plurality of meanings in the context of user-centric privacy-enhanced identity management as discussed as part of the efforts related to the PRIME project [ACC⁺05, CPS11]. We discuss the different dimensions of trust, ranging from the social to the technical dimension, as well as its temporal, risk, delegation, and dynamic aspects. Those discussions equally play a role for the system and framework presented in this thesis, which is an evolved subsystem of the one elaborated within the PRIME project.

1.5.3 Orthogonal Contributions

The results presented in this thesis and the complementary results are targeted at defining a comprehensive framework and system for privacy-preserving identity management using private certificates, thereby resolving systemic challenges of such system. The thesis is only focused on this narrow, yet important, topic, while, in order to comprehensively protect the privacy of citizens, further areas need to be addressed outside of the thesis scope. To this end, further contributions to topics related to privacy and security have been made by the author in the course of the research in the field of privacy. As those results address privacy from different standpoints than our framework and system, they can thereby be seen as orthogonal results. The areas touched by those results are such where privacy is at risk to a high degree: social networking, virtual reality systems, and identity resolution. Next, we present an overview of those additional contributions, their value to privacy, as well as how they relate to our core results of the thesis by addressing a more comprehensive view on privacy.

Access control in electronic social networks. A privacy protection issue which is completely orthogonal to that of minimizing data to be released by a party in a transaction has come up with the widespread use of electronic social networks. An electronic social network (ESN) holds profile data of users and it is the very purpose of an ESN that those data be available to other users. That is, data minimization towards the service provider is not applicable in today's service-provider-centric architecture models for ESNs—the service provider learns, by definition of the architecture, all profile data elements of all users. In this model, the goal for privacy is that the profile data of a user is available to other users only as decided by the holder (owner) of the profile. However, with the growing number of contacts a user has, it becomes an increasingly unrealistic assumption that the user is able to reasonably define the access control function for her profile data. A part of the definition of such access control function realizes the concept of *audience segregation* for social networks as discussed by van den Berg and Leenes [vdBL10, vdBL11].

We propose a method for access control to profile data which is based on users assigning *ratings* with respect to metrics on other users in a generic manner in Müller and Sommer [MS12a]. Those ratings act as inputs to an access control function which is used for computing decisions on which profile data items to make available to an ESN user. For example, if a user rates another user as highly competent in a certain area, this may lead to the other user gaining access to high-value business contacts of the user. This approach realizes user-defined access control to profile data the user holds in a generic manner, implying also the capability for realizing audience segregation as a special case. Thereby, the approach follows the principle put forth by Westin [Wes67] of users deciding on which data to release to which parties.

Further work we did in the area of electronic social networks relates to automating certain aspects of relationship management [MS10a], exploitation of contextual information in social networks [MS10b], negotiable information access [MS12b], or aspects of data integration of social networks [SM09].

Trust through electronic social networks. One main problem of the so-called Web of Trust [Zim95] public key infrastructure is that it depends on the ability of its users to ensure the association between public keys and their owners. Such association can, for example, be achieved by signing public keys once their binding to persons has been checked in physical meetings, e.g., through verifying credentials such as identity cards or passports of the claimed key holders. Such procedure is rather time consuming and not suitable for the average user, which may be one of the main inhibitors of the success of the Web of Trust PKI. We propose mechanisms for simplifying such trust management by utilizing electronic social networks (ESNs) as they already comprise a rich set of identity and trust information. We thereby build on dynamically changing profile information and interaction between users of social networks. In our model, we consider both trust establishment between parties as well as trust management of established trust relationships over time as relations change over time. The basic approach has been originally proposed by Müller and Sommer and published later [BMP⁺09]. The idea of exploiting electronic social

networks for trust management has also been discussed by Hogben [Hog09] in a parallel effort, though our work takes the ideas further and to a greater level of detail.

Our approach of leveraging electronic social networks for the management of the Web of Trust PKI can allow, once implemented using appropriate usability concepts, average users to leverage their ESN profiles and data to manage their Web of Trust public keys. This can allow one to overcome the non-acceptance of such PKI as of today, caused through today's tedious and complicated management processes for the users and thereby help move towards encryption and signing of electronic communication of users. This can substantially contribute to privacy of users by addressing communication privacy as well as better accountability of communication, should this be required. Again, this is orthogonal to the privacy benefits of the core part of this thesis. However, it also touches on important and still unresolved privacy and accountability issues of today's deployed systems of electronic communication. Realizing such idea is possible through extensions of social network platforms, either in their core or through applications, and does not require the interplay of a plurality of different players.

Privacy-enhanced authentication in virtual worlds. At the time when virtual worlds have gained strong public attention, we have investigated the integration of privacy-enhanced authentication in virtual worlds [BCH⁺09, BHSS10]. As virtual worlds are typically based on a centralized architecture such that the provider learns all information being exchanged in the virtual world, a main idea was to detach the authentication infrastructure from the virtual world and maintain it as part of a separate authentication infrastructure, connecting to the virtual world over simple interfaces. Another main idea was to allow real-world credentials, e.g., an electronic identity card, to be used in the virtual world, e.g., for age verification. The reasoning behind this is that in many use cases (certified) attributes of the human behind the avatar, that is, secure real-world identities, are relevant for an authentication of the avatar in the virtual world. Consider age verification in the virtual world as an example for this. The possibility of the use of real-world credentials is implied by integrating a standard authentication infrastructure with the virtual world. The proposed architecture ensures that the virtual world platform provider learns the minimum amount of information with respect to authentication transactions of the users.

The feasibility of the ideas has been shown through a prototype implementation of the architecture based on our privacy-enhancing authentication architecture and prototypes [Som07, CGS06, CSSZ06, CSSZ07] used within the Second Life⁵ virtual world platform as an authentication mechanism outside of, and integrated with, the virtual world system.

A further item of work relates to privacy-enhanced presentations in virtual world infrastructures [BCG⁺10], which allows an authenticated group of parties to attend a presentation and view it in a confidential manner on a public screen, while non-

⁵<http://secondlife.com>

authenticated parties see public image material on this screen.

Privacy-enhanced database matching. Contributions in the field of privacy that are orthogonal to our privacy management system have been made in the area of *identity resolution*, which is the processing of multiple databases for identifying entries of different data sets that match in terms of specific criteria. Thereby, fields of data records of different databases (from different data suppliers) are compared with each other to identify similar records that probably belong to the same person. Such systems are used, e.g., for identifying fraud when a single person uses multiple (similar) identities, to consolidate customer databases, or for threat detection in the national security context. Our work was performed as a possible extension of a product in this area from a major IT services company to better protect personal data in the identity resolution context.

The simplest approach is that the matching party, or matcher, obtains all to-be-matched databases and runs the processing on those local copies. This, of course, reveals the complete databases to the matcher. An extension of this basic scheme is based on computing salted cryptographic hashes of data items which improves on the basic scheme but is susceptible to phone book attacks through which it is not hard to reconstruct large portions of a database. Products implementing such schemes are available and used in practice, both by governmental and private organizations. Overall, the idea of applying identity resolution schemes counters the idea of data minimization because of the potentially excessive sharing of user data between parties. Though, as this is done in practice with mechanisms worth improving, we investigated this field and proposed schemes with better privacy protection. Thereby, we contribute to a field orthogonal to our main part of the work in order to improve on users' privacy from a more comprehensive perspective.

We improve on currently implemented identity resolution technology by proposing a family of schemes in which all parties contributing a database to an identity resolution process need to acknowledge all data of all other data contributors, where they obtain the data in an obfuscated form only. This effectively prevents phone book attacks of previous schemes that uncover large parts of the databases and also allows for quantitative control of the identity resolutions being done by the matching party. We provide a system architecture as well as a proposal for a realization of what we refer to as *commutative obfuscation function*, of which commutative encryption can be an instantiation. On top of our basic scheme we propose a number of extensions for further strengthening privacy protection of the databases resulting in a family of privacy-enhanced identity resolution schemes [VPCS11a, VPCS11b]. Privacy-enhancing identity resolution may reduce the disclosure of identifying personal data while retaining utility of the data in the private sector context. In the national security domain, it is expected that unobfuscated personal data will be preferred.

1.5.4 Summary of Contributions

Summarized, a major contribution of this thesis is to show how the theory in the form of cryptographic protocols can be transferred towards practice⁶ through a system design and framework that is comprehensive in multiple dimensions. The overall work has started with a vision of privacy-preserving identity management and the basic cryptographic protocols. We present a comprehensive architecture for a system that comprises data representation, authorization, reasoning over ontologies, and cryptographic protocols to realize this vision. All those aspects have been developed to be aligned with each other for obtaining a coherent system. We have realized multiple prototype implementations of selected aspects of our work for validating our approaches, e.g., a version of the credential protocols with a first set of features targeted at practical deployments, an integration with standards thereof, and a user interface, and shown through a complete showcase that application of the protocols for practical Internet authentication scenarios is feasible today. This fully functional identity management showcase based on our implementation results has integrated all implementation results at the time for obtaining a complete system. In this respect, our results fill substantial gaps in the domain of privacy-enhanced identity management that have not been addressed before, particularly also in making the cryptographic protocols practical. We have overcome main challenges related to integrating the complex cryptographic protocols into a system, while exposing their semantics through a logic-based language.

In addition to those core results related to the thesis, we have obtained complementary and orthogonal contributions in the domain of privacy-enhanced identity management as discussed above. Jointly, the contributions cover the system for data-minimizing identity management as well as related domains crucial for the protection of user privacy deeply and cover the wider problem domain of privacy and security with further selected results in key practical problem domains.

1.6 The Need for Data-minimizing Authentication for the Future Internet

Considering the rapidly evolving personal data economy, the potential cost, in both monetary terms and hassle, of data breaches, the excessive release of data by users for performing online transactions, and the direction of regulatory initiatives in both Europe and the United States as discussed in Sec. 1.1 above, we conclude that there is a strong need for data-minimizing authentication on the Internet. We see such technology to be used in conjunction with tools for allowing for user control of profiling and network-level unlinkability, the (semi-)automated agreement of data handling policies with service providers, and an automated preferences-driven approach for applying those technologies in a concerted manner. Using such concerted approach for controlling both explicit as well as implicit release of data

⁶This is made explicit in the subtitle “From Cryptography to Practice” we have chosen for the thesis.

can help balance the privacy interests of the users with the economic interests of service providers and third parties, while keeping services free of monetary charge. This is, in our view, a likely scenario in the future. Clearly, users should have the possibility of choosing to be fully untraceable for certain interactions, while still being capable of asserting certified attributes, when they feel sensitive personal data is involved and needs to be protected. Otherwise, a certain (user-controlled) amount of data release is necessary for personalization of services and targeting of advertising. Thus, our proposed privacy-enhanced authentication system provides a strong solution to parts of the privacy problem space, helping users get better control over their personal data release.

Both Europe and the United States have launched government-funded research programs for progressing the state of the art in privacy-enhanced identity management. The European PRIME program [PRI08] has addressed data minimization in identity management in the direction of this thesis, while the PrimeLife program [Pri10] has taken a broader perspective, particularly also taking new challenges resulting from user-generated content, particularly social networking, into account. A more recent initiative in Europe is the ABC4Trust project [ABC13] which pilots privacy-enhanced authentication technology based on credentials. A notable and more recent initiative in the USA is the NSTIC program [The11, Nat10], which also addresses data-minimizing authentication as one of its topics, while piloting the technologies. This further underlines the interest of major governments in privacy-enhancing technologies like those proposed in this thesis.

The author has made substantial technical contributions to the PRIME program and managed the PrimeLife program at the time he was working on this thesis. The work in PRIME has lead to the earlier results and publications related to the thesis and has resulted in requirements for the later refinement of the results presented in the thesis.

A potential game-changing paradigm shift in the future towards a market for personal data in which users trade personal data against monetary payments could radically change the personal data economy of today and put the user into an overall stronger position. In case of such changing paradigm on how personal data gets handled, using our privacy-enhanced authentication approaches and other identity management techniques may give those willing to pay for privacy close-to-perfect privacy, while users who want to earn by having their (personal) data exploited, let companies use them at their discretion.

1.7 Thesis Outline

After this introduction in the current Chapter 1, we first provide an introduction to terminology and concepts in Chapter 2, for this being instrumental for the understanding. Thereafter, we present the secure channel model, with a focus on authentication in Chapter 3. In Chapter 4 on the data model we go into details on how data and requests are represented in a formal calculus and discuss ontology-based deductions for the authentication context. In Chapter 5 we apply the data

model to specifying the interfaces of our cryptographic protocols. In Chapter 6, we give an overview of our run-time generation framework for cryptographic protocols. Due to space constraints, a full specification thereof cannot be given in this thesis. In Chapter 7, we summarize the implementation efforts related to the results in the thesis. We conclude in Chapter 8 with future work and an outlook on the deployment of privacy-enhanced authentication technologies.

Chapter 2

Concepts and Terminology

In this chapter, we introduce basic terminology and present some general yet important concepts for privacy-enhanced identity management which we will build upon and adapt towards our needs. Thereby we build on the terminology presented by Pfitzmann and Hansen [PH, PH10] and adapt or extend it where necessary. We recommend their work to the reader as an excellent and comprehensive introduction to privacy terminology with a focus on data minimization and privacy.

2.1 Basic Terminology

An *identifier* is a bitstring or name that refers to a party in the system. An identifier can reveal attribute information about a party, e.g., the legal name of a party, or can be semanticless, as is the case for a pseudonym. Attribute statements can be associated with an identifier, without the referred-to party necessarily being identified through its legal identity. A special kind of identifiers are *pseudonyms* as introduced further below.

The *civil identity* or *legal identity* of a party comprises the attributes of the party that have been assigned to it officially, e.g., through governmental institutions. Examples for attributes comprising legal identities are the first name, last name, date of birth, or social security number. Attributes such as the passport number and identity card number as contained in concrete credentials of the party are further

examples of attributes of the legal identity.

A *transaction* between parties is a sequence of messages exchanged between the parties to accomplish a given task. In this thesis, we are mainly interested in transactions for establishing authentications between parties. A transaction does not have the strong properties, e.g., atomicity, of transactions in the database world.

A party is *identifiable* within a set of parties, the *identifiability set*, if the party is uniquely characterized through the knowledge available about it to an *observer*.

Anonymity of a party is defined as the party not being identifiable by an observer within a set of parties, the *anonymity set*. The anonymity set is defined through the information known about the party and the distribution of attributes among parties.

A party is *pseudonymous* if it uses a pseudonym identifier for referring to itself in a transaction with another party, or if another party uses a pseudonym identifier for referring to this party. Pseudonymity does not make a statement about the anonymity of the party.

Unlinkability of two transactions is defined as the observer not being able to decide whether the same party has been involved in both transactions. When considering unlinkability, the observer is often one of the parties involved in the transaction or a coalition of multiple such parties. *Linkability* of two transactions, on the contrary, means that the observer has information allowing it to deduce that the same party is involved in both transactions.

The observer referred to above can be any party from the perspective of which the above properties are to be assessed. Note that a common case when considering privacy is that the observer is comprised of a coalition of multiple parties in a system, e.g., a service provider a user authenticates to and involved certifiers of identity attributes. The identifiability and anonymity properties are always relative to the observer which we consider. When we mean an observer with malicious intentions, we may refer to it as an *attacker*. An observer is a passive attacker, while an active attacker may execute actions not corresponding to the agreed protocol.

A party's civil identity attributes typically make a party identifiable considering the identifiability set comprising all players in the system, while a pseudonym without any further attribute statements associated with it, will usually not make the party identifiable, but let it remain anonymous w.r.t. the anonymity set comprising all parties in the system. When a party uses the same pseudonym in two transactions, those transactions are trivially linkable.

Identity management in this work is following a wider definition than that in Pfitzmann and Hansen [PH10], which is essentially constrained to the management of a party's partial identities comprising the development of partial identities and choice of such and pseudonyms for use in a transaction. The notion of identity management used in this thesis takes a broader perspective and covers all aspects of handling digital identity data throughout their life cycle by all parties involved. This comprises particularly, but is not limited to, a party's (user's) management of partial identities in the sense of the related earlier work [PH10], the release of identity data to recipients including user interactions, aspects related to authorization to resources of a party through release of identity data, and the enforcement of agreed policies for

handling released data. This matches the prevalent notion of identity management in the literature body [CLHS11, HBC⁺04, Koh10, PLFK11, PKK⁺11]. The identity management system presented in this thesis focuses on the data-minimizing release of attribute statements, including the handling of credentials, policies, and user interactions. The system can be seen as part of a larger system with additional functionality as that proposed in the PRIME project [PRI08, Som11].

Another part of the literature views *identity management* from the enterprise perspective as the management of identity data and related identifiers of parties, e.g., employees and customers, relevant in the enterprise context. The common meaning of the term is aligned with this enterprise-centric notion [Bla11] rather than with the research notion adopted in this work.

Privacy-enhanced identity management is identity management that can realize data minimization, that is, facilitates unlinkability of transactions where desired, and keeps partial identities minimal. In a broader scope, such system provides further mechanisms for protecting user privacy, e.g., a policy-based data life-cycle management subsystem. *User-centric identity management* refers to the approach of allowing the users to have control over their identity data and the use thereof. *Privacy-enhancing user-centric identity management* is the combination of both above concepts of user-centric identity management and privacy-enhancing identity management and is the approach followed in this thesis. [BSCGS06, BSCGS07]

2.2 Data Minimization

The concept of *data minimization* refers to constraining the collection of personal data by a data controller to the minimum required for a specified purpose. It is crucial that this minimum is to be seen relative to the purpose the data are collected for—according to the closely related purpose specification principle, data may only be collected by a data controller for a specified *purpose*. Following the definition given in related work [PH10], data minimization also encompasses that the mere possibility of collecting personal data about others be limited as well as the temporal dimension, that is, that the time that collected data is stored by the data controller be minimized.

The data protection principle of data minimization has its legal roots in the early foundations of European and international data protection, namely the European Convention on Human Rights (ECHR) [Cou50], the Data Protection Convention of the Council of Europe⁷ [Cou81], the OECD guidelines on data protection for transborder data flows [OEC80], and the European Data Protection Directive 95/46/EC [Eur95]. The principle has not been referred to as data minimization principle explicitly, rather it can be derived from other principles—the *purpose specification* principle and the *proportionality* principle. The purpose specification principle, also known as finality principle, can be derived from Art. 6.1(b) of Directive 95/46/EC [Eur95], Article 5(b) and (c) of the Convention 108 and Art. 8 of the European Convention on Human Rights. The proportionality principle has been put

⁷This is the *Convention 108* of the Council of Europe and often referred to as such.

forth in Art. 6.1(c) of Directive 95/46/EC and Art. 5(c) of the Data Protection Convention of the Council of Europe [Cou81]. Article 6.1(e) Directive 95/46/EC implies that data may be kept only in anonymized form or must be deleted once not required for the purposes for which they have been collected. The principle of data minimization is denoted as such explicitly in some data protection acts that implement the Directive 95/46/EC, e.g., Article 3(a) of the German federal data protection law (Bundesdatenschutzgesetz (BDSG)) [Deu03]. See, e.g., Schnabel [Sch09], for a discussion of the basic legal principles underpinning data protection in Europe.

The guidelines and legal regulations consider the general notion of collection of data by the data controller. The setting of the release of certified identity data relevant in this thesis is a special case of this and data minimization becomes mainly relevant in the context of a party, the user (data subject), releasing data to another party, the service provider (data controller). Also worth considering in this context are the certifiers facilitating the release of certified data, to which, when interpreting the guidelines and regulations in the light of the technology used, the principle of data minimization should be applied equally. We leave the discussion on whether certifiers are data controllers or only data processors in a data release transaction as an open issue for the legal domain. Using such interpretation, private certificate technology is the only technology capable of realizing data minimization in the setting of certified attribute information as it prevents certifiers from learning information about a user's transactions and prevents service providers from learning excessive information through the intrinsic properties of other certification technologies.

Data minimization is the key privacy property we target to achieve with our approach for the release of certified data which constitutes the main part of the thesis. The notion of data minimization we adopt is the one of data minimization towards the data recipient (data controller or data processor acting on behalf of the data controller) and also towards third parties that may be involved in the transaction of releasing data. We are not particularly interested here in data a data controller has obtained from different channels than the release by a user—this is part of a larger architecture and system, aspects of which are, e.g., discussed in related work by the author [Som11].

Data minimization is a concept that, if it is to be realized in practice, does not only require support at the technical level which we are discussing in this work, but also at the level of business processes that need to be (re)defined accordingly to operate on the minimum amount of data possible. Also legal considerations come into play when discussing data minimization, e.g., the consequences in the case of a dispute related to anonymous transactions, or whether anonymous interactions are legal in certain jurisdictions. The non-technical issues are equally important as the technical ones for a practical deployment and are treated in the corresponding literature.

2.3 Pseudonyms

Our model is, as mentioned already, targeted at interactions between parties where parties are not necessarily revealing their legal identities to each other. They rather use identifiers that do not comprise any other attribute semantics than being an identifier, or name, for the party in a specific interaction with another party. Such an identifier is used as a mutually known reference to the subject this identifier refers to. Thereby it can serve as a reference to the party used by one or more other parties. A party can have many such identifiers, particularly also multiple ones with a single other party, and can control by itself whether it wants to link any of those or keep them unlinked in the view of the other party or another observer. We call such identifiers *pseudonyms* or *subject identifiers*.⁸ The term pseudonym has its roots in the ancient Greek term “pseudonumon” which means falsely named.

Definition 2.1 (Pseudonym) *A pseudonym is an identifier of one specific party A that is established with one, multiple, or all other parties in the system.*

A pseudonym can refer to the party who uses it which is the most prominent case for pseudonyms, or it can refer to other parties which is, for example, required for referring to a third party in a delegation interaction or for a service provider making statements about data about one of its customers to another service provider while using different names for the customer for different parties it talks to or in different interactions. Our concept of pseudonyms thereby generalizes the notion of pseudonyms in the identity management literature where a pseudonym is defined as being an identifier of the party that uses the identifier [PH10].

Pseudonyms have been introduced by Chaum, who referred to the concept of a digital pseudonym as a public key with respect to which holdership of a private key can be proven through a digital signature [Cha81]. The application of pseudonyms together with credentials has been first discussed in Chaum’s later publications [Cha85, Cha86, Cha90].

Certain scenarios merit from a party having a single pseudonym it uses in communication with all parties. As an example, a well-known service provider or certifier on the Internet may benefit from, or even require, having only one pseudonym and it does (and need) not take advantage of the privacy that multiple pseudonyms offer. Such pseudonyms are denoted as *public pseudonyms* and are commonly used for referring to parties acting under their legal identity at all times, such as service providers, certifiers, or trustees.

Different kinds of pseudonyms have been discussed in the literature, with different properties in terms of unlinkability of transactions referring to such pseudonyms, namely person pseudonyms, role pseudonyms, relationship pseudonyms, role-relationship pseudonyms, and transaction pseudonyms [PH10]. This classification depends on the contexts a pseudonym is (re)used in and affects linkability. For a

⁸Note that the term identifier does not imply that any attribute information about the party’s civil identity be known. The purpose of an identifier is only to distinguish the party within a given context.

detailed discussion of pseudonyms and related concepts, we refer the reader to the already-mentioned seminal paper [PH] and its extended version [PH10].

2.4 Credentials

The term *credential* in the context of privacy-enhancing identity management has been introduced by Chaum in his seminal paper on credential systems [Cha85]. Chaum has considered a credential issued to a pseudonym of a party by a certifying organization (i.e., certifier) to be a statement by this organization that can be transferred under a different pseudonym of the party to another organization in an untraceable way. The statement was determined through the kind of credential being issued, and no attributes could be endorsed. Expressed differently, a credential is an object that allows its holder to prove its relationship with one organization to another organization in an untraceable way. The ideas presented in this early paper are the basis for later practical approaches to credential systems, though, no practical system was defined there. Chaum presented a first cryptographic implementation of a credential system in his later papers [Cha86, Cha90].

Lysyanskaya et al. [LRSW00] provide a formal definition of pseudonym and credential systems, a theoretical construction based on any one-way function, and an efficient and practical scheme for single-use credentials. Camenisch and Lysyanskaya [CL01] define a first practical scheme for credentials with multi-show unlinkability, that is, a credential can be shown a polynomial (in a security parameter) number of times and all those transactions are unlinkable to each other and the issuing transaction. Their work is the basis for the *Identity Mixer* or *idemix* credential system which is the basis of the protocols presented later in this thesis. For this thesis, we define credentials in a more generic manner as a set of typed attributed with their values and a signature on those.

Definition 2.2 (Credential) *A credential is a cryptographic certificate comprising typed attributes and their values, issued on a pseudonym of a party to this party by a certifier (issuer). Holdership of a credential can be proven to other parties by its holder under different pseudonyms than the one it was issued on. Such proof of holdership means transferring the credential from a pseudonym with the issuer to another pseudonym with the recipient party and optionally revealing parts of the attribute information contained therein. The credential transfers are, like in the original definitions, unlinkable to the credential issuing transaction.*

This non-formal definition of credentials is aligned with the formal definitions of [LRSW00] and [CL01], while in addition considering attributes. In this thesis, we also refer to credentials as *private credentials* or *private certificates*. Those names emphasize that the credential is not released in a transaction of transferring it. We may also refer to them only as *certificates* when it is clear from the context that we mean credentials.

2.5 Partial Identity

A *partial identity* has originally been defined as a set of identity-related attributes of a person, where an attribute is the pair of attribute name and value [PH, PH10]. A partial identity thus exposes a certain facet of the identity⁹ of a person and the person should be in control of defining and using partial identities in interactions with other parties at her own discretion. That is, a person can decide which partial identity to expose to another party in an interaction. See again the related seminal work for a discussion of partial identities and related concepts [PH, PH10].

As we take the approach of data minimization even further than envisioned in previous work on the subject [PH, PH10], we also generalize the above definition of the concept of partial identity of a subject to take stronger ideas in terms of data minimization that we employ in this work into consideration. The main generalization is that logic formulae define a partial identity, not only sets of attribute-value pairs.

Definition 2.3 (Partial identity) *A partial identity of a party \mathbf{A} is a set of statements about \mathbf{A} , where a statement is a formula comprising assertions about \mathbf{A} 's attributes, expressed through predicates in a formula comprising the logical connectives \wedge and \vee with their standard meaning.*

Details on how those statements can be expressed in a formal language are presented in Chapter 4. Like in the original definition, a partial identity exposes a certain facet of the identity of a subject (party) and the party should have control over defining (controlling) her partial identities with other parties. Our extension of the definition accounts for our requirement of expressing predicates over attributes in addition to only attribute values as well as disjunction as logical connective to take data minimization even further without leaving the domain of efficient cryptographic protocols for releasing such data-minimizing statements.¹⁰ Using such features allows a user to define a partial identity based on any such statement, thereby preventing certain unnecessary releases of attribute values where this is not required for the purpose of the data release. Our changes retain the conceptual ideas of the notion of partial identity and can be seen as an extension thereof for obtaining better data minimization by using formulae making statements about identities to specify the data about a subject instead of only sets of attribute-value pairs. The concept of conditionally released identities as introduced in Sec. 4.4 also contributes to the concept of partial identity in the sense of comprising information that a data recipient or third party may obtain under certain conditions at a later point. Both conditionally released and opaque identities may, if used within a formula, reveal, through their relations

⁹The term identity is used here as the totality of attributes of a party, in contrast to the predominant meaning in this work of an identity being a named set of tuples, each tuple representing information about an attribute of the identity, as introduced in Chapter 4.4.

¹⁰Note that an attribute predicate can always be expressed as an attribute reflecting the predicate, though, this approach is conceptually not clean as it unnecessarily creates new attributes “hardcoding” the predicates for emulating those. Thus, this is not a practical approach in the general case and particularly not a substitute for our extensions.

to other identities, information on which third-party endorsed attributes the party holds credentials for.

Note that the concept of *identity* that we introduce in Sec. 4.4 deviates from the well-established term of identity being the set of all identity attributes of a person. An identity in our definition is rather a set of tuples of attribute name, attribute value, and data type certified by a certifying party. Any entity can be holder and subject of an identity, where entities can, among others, be natural persons, legal persons, or data processing systems or services.

2.6 Accountability

Accountability is a term that is hard to define precisely and that has contextual meaning. It is often used in the public and private sector governance contexts, where the common notion of accountability of a party towards another party is that the accountable party must give account to and is liable for its actions towards the other party, which is also well-aligned with the common meaning of the term.

In this thesis, we are interested in accountability of parties executing electronic transactions with other parties, such that the parties can be held responsible and liable for their actions towards the other parties in case of misconduct. We also refer to accountability of a transaction, with the meaning that sufficient evidence is created in the transaction which allows to hold the party executing the transaction in case of misbehaviour liable towards the other party involved in the transaction. For the case of a transaction using delegation, both the delegatee and delegater may be accountable towards the other involved party of the transaction. Following this, we define accountability with respect to a transaction between an authenticating party **A** and a verifier **B** as presented in Definition 2.4.

Definition 2.4 (Accountability enabling, accountable) *A transaction between an authenticating party **A** and a party **B** is accountability enabling if sufficient evidence is established at **B**'s side which allows **B** to obtain, once a pre-agreed condition c becomes fulfilled at a later point in time, a set of typed attribute-value pairs about **A** and/or another party **D**, authenticated by **A** and certified by one or multiple identity providers. The attributes allow to identify **A** or **D** in a given context in order to establish the possibility to hold it or them responsible or liable for its or their actions. Parties **A** and/or **D** are accountable towards **B** in the sense that, once the condition c becomes fulfilled, **B** can identify **A** and/or **D** and hold it/them responsible and liable through external mechanisms.*

Accountability is crucial in the context of compliance with legal regulations and protecting a service provider's interests against the interests of dishonest users. Thereby, it allows for establishing a balance between the stakeholders in the system. Our system achieves accountability in a setting where parties need not be identified w.r.t. (parts of) their legal identity in order to be accountable.

A formal definition of accountability in the context of cryptographic protocols has been presented by Küsters et al. [KTV10], which defines accountability as the

property of a protocol that, if a desired goal of the protocol cannot be met due to misbehavior of participants, those participants can be blamed. This definition captures only the cryptographic protocol, though, in a very generic manner, while our informal notion is more appropriate for the context of privacy-enhancing identity management.

2.7 Unconditional and Conditional Release

When a party **A** releases (parts of) an authenticated data statement to another party **B**, this release can be either *unconditional* or *conditional*. When data are released unconditionally, they are learnt by the recipient upon or before completion of the corresponding release protocol. Conditionally released data, on the contrary, are only learnt by their recipient once an agreed condition—the *release condition*—becomes fulfilled.

Definition 2.5 (Unconditional release) *Unconditional release of data by a party **A** to a party **B** is defined through party **B** learning the data at the latest upon successful completion of the release protocol.*

Unconditional release is the common case for releasing data and used in most of today’s transactions when data are revealed by parties to data recipients.

Definition 2.6 (Conditional release) *Conditional release of data by a party **A** to a conditional data recipient **T** under an agreed release condition **c** means that **T** only learns the data once and if **c** becomes fulfilled after completion of the release protocol.*

Conditional release can be used to realize accountability of **A** towards **B** by conditionally releasing attribute information about **A**, allowing **A** to be identified in a certain context if the condition is fulfilled, in an otherwise pseudonymous or anonymous interaction.

The concept of conditional release has been introduced in the context of credential protocols in the framework of Bangerter et al. [BCL04]. Using cryptographic mechanisms, this concept can be realized in a strong trust model [BCL04]. Concretely, it can be realized through verifiable encryption [CD00, CS03], where **A** encrypts certified attributes under the public encryption key of **T** and the condition **c** as a *label* of the encryption, releases the ciphertext to **B** with which it executes a data release protocol, and proves—as part of the protocol—that the ciphertext contains the certified attributes as stated.

Multiple trust models can be overlayed over a verifiable encryption protocol for enforcing the conditional aspect of the release: Either or both parties may be trusted to enforce the condition **c**. When trusting party **B** to enforce the condition, **B** will only forward the ciphertext to **T** once **c** is fulfilled, otherwise it may forward it immediately. When trusting **T** to enforce the condition, we can assume it will only decrypt a ciphertext it receives once **c** holds. In either setting, a third party, e.g.,

a law enforcement agency, or, if **B** is not the only party assumed to be trusted for enforcing c , **B** can be the ultimate recipient of the data. A typical setting is that **B** has no incentive to send the ciphertext to **T** unless c holds, and **T** is trusted for enforcing the condition.

For settings where **T** is trusted for enforcing the condition, we refer to it also as *conditional-release trustee*, or *trustee*. The party needs to be trusted by the authenticating party that it will not allow another party or itself to learn conditionally released data before the condition is fulfilled. And it needs to be trusted by the verifier that it will allow it to learn the conditionally released data once the condition is fulfilled.

2.8 Delegation

In practical information systems, it is often a requirement that parties can receive the right to act on behalf of other parties. We next present a generic definition of delegation that is aligned with common definitions.

Definition 2.7 (Delegation, delegater, delegatee) *Delegation is defined as a party, the delegater, giving a subset of its rights it holds in a system to a party, the delegatee, who can then exercise those rights on behalf of the delegater.*

We next present the traditional model for delegation and the delegation of attribute authority. Those approaches to delegation differ fundamentally in terms of how the delegation is achieved and the identifiability requirements of parties, or, stated positively, the possibility of parties of remaining anonymous or pseudonymous, only known under attribute statements.

Traditional delegation model. In a system where parties are identified, delegation can be realized through an access control policy capturing a delegation of actions or a role for granting the delegatee access to a resource in addition to, or in place of, the delegater, based on the delegatee's identifying attributes or a unique identifier. Such traditional approach to delegation is thus realized through authorization policies and the authentication system can remain largely unchanged and does not need to reflect delegation. This approach is conceptually well aligned with the paradigm of parties being identified towards each other with their civil identities or unique identifiers. This works well in systems where no credentials for proving holdership of attributes, capabilities, or roles are available, but users authenticate always with respect to their unique identities and authorization is based on this. A plurality of services offered on the Internet today that require authorization follow this tradition. Use cases for this traditional delegation approach are numerous and not elaborated in this work.

Delegation of attribute authority. Though, for the more general model of attribute-based authorization and access control we build on for our work, delegation can be extended to the delegation of authority over the use of a delegater's (certified)

attributes to a delegatee. The delegatee can authenticate attribute statements to verifiers on behalf of the delegater and get authorized to perform actions on the verifier's behalf. This realizes the model of delegation in a system where parties are not necessarily known under a unique identifier or their legal identity, but known under pseudonyms and attribute assertions.

This approach for delegation allows the delegatee to use the delegated attributes as capabilities for gaining access to a resource. A special case is the pseudonymous delegation of roles such that the access rights linked to a role are delegated with the role, where roles are modeled through attributes. This is a generalized embodiment of role-based access control compared to an embodiment based on traditional identity schemes and identified users because in our scheme both the delegater and delegatee can remain pseudonymous or anonymous to the service provider.

Any use case where a party or at least the delegatee need not be identified are suitable for this approach to delegation. Use cases where both parties should be identified can be handled with the traditional delegation model as a trivial case of our system, the delegation being handled on the authorization layer.

Example 2.1 (Delegation) *As a simple example, the holder of an access credential to an online resource, e.g., a commercial database, can delegate this credential to a colleague to have actions related to the resource performed by this colleague, possibly for a limited time duration. Thereby, neither of the parties needs to be identified to the service provider, which is a main privacy advantage of our approach. Concrete examples are access to a patent search database delegated for parts of a project or the booking of a trip by a secretary for her manager where the secretary can remain pseudonymous to the travel booking service and only prove her eligibility of acting on behalf of the delegater without releasing a unique identifier or identifying attributes.*

Chapter 3

Authentication Model

Following the motivation for the need of privacy-enhanced identity management given in the introduction, we present our model for privacy-enhanced authentication in this chapter as part of a secure channel model. Authentication in the sense of third-party corroborated attribute statements about parties being conveyed to a verifier can be considered a crucial aspect related to digital identity management because authentication poses severe privacy risks if not done with appropriate provisions for privacy.

3.1 Introduction

The model presented in this chapter expresses the credential-based identity management approach that is at the core of this thesis and the system for identity management that we define. The main goal of this system is the establishment of *authentication relationships*, or *authentications*, between parties. Informally, we mean with authentication that authenticity of a statement about a party, and possibly other parties, be established towards a recipient.

Using the concepts and terminology of the secure channel model and algebra of Maurer and Schmid [MS94, MS96] and its recent extensions by the author [BCS12a] and the model presented in this chapter, the main goal of our system is the establishment of *authenticated channels* between parties. Authorization policies are not considered explicitly in the channel model we propose and it is assumed that parties

authenticate attribute statements such that the recipient policies are fulfilled for reaching the goal, e.g., accessing a resource.

The authentications to be established between parties in the system are governed through the *authorization policies* of the parties. An authentication of a party with another party is always established for a well-defined reason. Abstractly, this reason can be modeled as an access to a resource of the party by the other party with authorization policies being specified over this resource and governing the required authentication.

We present the related work for the channel model in Sec. 3.2. Thereafter, we present the Maurer–Schmid model [MS94, MS96] for secure channel composition in Sec. 3.3 as a prerequisite for the later work. Based on this and our previous work on channel composition models by Bichsel et al. [BCS12a], we introduce our extended secure channel model in Sec. 3.4 and thereby keep a focus on the authentication property. We present a set of transformation rules of a channel composition algebra for our extended model in Sec. 3.5. We explain the separation of authentication and authorization functionality for our system in Sec. 3.6 and elaborate on the concept of evidence for authenticating statements in Sec. 3.7. We discuss in Sec. 3.8 how one can factor trust in parties out of the authentication layer and consider it in the orthogonal authorization layer and we elaborate on the concept of registration of a party in a system in the context of the model and its importance for bootstrapping an identity system in Sec. 3.9. We conclude the chapter with thoughts about future work in Sec. 3.10.

3.2 Related Work

In the area of modeling security of authentication and communication channels, numerous recent papers are available. Typically, they focus on formally verifying security properties of protocols in an automated fashion. Backes, Maffei, and Unruh [BMU08] have integrated zero-knowledge proofs, a major building block of privacy-friendly authentication, into an automated verification tool. Mödersheim and Vigano [MV09b] have later put forth a formal model of pseudonymous channels. Following their approach of modeling pseudonyms, we could aim at a more formal model of attribute-based statements and conditional release of information. However, we want to focus on the intuition and consider a rigorous formalization to be an interesting future contribution. Notable work addressing pseudonymous authentication channels that provides a formal model and tool-based verification of a subset of the Identity Mixer protocols has been published by Camenisch, Mödersheim, and Sommer [CMS10].

Regarding semi-formal models of authentication, Maurer and Schmid [MS94, MS96] have introduced a simple, yet expressive, notation allowing for analyzing and comparing protocols that establish secure channels based on standard cryptographic technologies available at the time. This model represents the starting point for various recent formal approaches towards modeling cryptographic functionality [Mau11, MRT12]. In our paper, we have the same goals as the original model,

but for substantially more complex protocols, the properties of which are harder to grasp and only understood by a small group of privacy or cryptography researchers. In contrast to the aforementioned proposals [BMU08, CMS10, MV09b] that use more complex and less intuitive notation for achieving their protocol specification and automated verification goals, our model extends the intuitive notation of the model of Maurer and Schmid while retaining its basic concepts and—partially—its simplicity. Our model can, like Maurer and Schmid’s, be used for comparing and analyzing security properties, especially for today’s authentication protocols. In addition, our calculus can act as a teaching model for the goals and properties of complex protocols and thereby contribute to a wider understanding of privacy-friendly authentication and accountability technologies and their future deployment. Therefore, our work closes the gap in the space of semi-formal models of expressing cryptographic schemes for privacy-friendly authentication with an intuitive yet formalized method. In this respect our work is orthogonal to the results in the space of formal protocol verification that have been presented before.

Our extension for attribute-based authentication requires authentication properties to be expressed in a suitable language. Sommer [Som11] presents a logic-based requirements language and dual specification language for attribute-based authentication supporting advanced schemes such as the Identity Mixer anonymous credential system. A closely related language for specifying attribute-based authentication requirements has been put forth by Camenisch et al. [CMN⁺10]. Both contributions allow for combining anonymity of transactions with user accountability based on the ideas originally put forth by Backes, Camenisch, and Sommer [BCS05].

3.3 Maurer–Schmid Channel Model

For our model of authentication, we build on the secure channel model proposed by Maurer and Schmid [MS94, MS96] and its extension by Bichsel, Camenisch, and Sommer [BCS12a]. Those works model communication relationships between parties as *channels* that can have certain security properties. Messages sent over such channels then are authenticated or confidential with respect to the properties of the channel. A channel transformation algebra defines transformation rules to obtain new channels with certain security properties from a set of available channels with certain security properties.

The choice of the Maurer–Schmid model as a basis of our earlier work [BCS12a] and the results of this chapter is founded by its simplicity and capability of modeling security associations between parties in an easy-to-grasp way. We use their model despite it not having a fully formalized association with cryptographic protocols realizing channel transformation rules.

Maurer and Schmid [MS94, MS96] have created their model for comparing the security properties of cryptographic protocols and reason about establishment of secure channels in open communication networks. Their model captures the security properties *authentication* and *confidentiality*. The transformation rules for obtaining new channels from existing channels are entirely based on traditional, in the sense

of non-privacy-enhancing, cryptographic primitives, such as message authentication codes, symmetric key cryptography, public-key cryptography, and digital signature schemes, existing at the time they devised their model.

Maurer and Schmid define channels to connect parties where parties can have security assurances regarding each other depending on the security annotations on the channel. Maurer and Schmid use the notation (3.1) for an insecure channel from **A** to **B**, (3.2) for an authentic channel where **A** is authentically known to **B**, and (3.3) for a confidential channel where **A** rests assured that its messages can be only read by **B**. A secure channel fulfills the authentication and confidentiality properties of the respective channel endpoints and is denoted as in (3.4). In their notation a bullet, a *security annotation* of the channel, denotes a security property, that is, either authentication or confidentiality of the respective channel endpoint. A security annotation on a channel means, in other words, that the party denoted as channel endpoint is the only party having access to the channel. For an authentic channel, this means, only the endpoint party with a security annotation can send a message over the channel, for a confidential channel, that only the endpoint party with a security annotation can receive a message on the channel [MS94]. In the channel notation used by Maurer and Schmid, a channel endpoint party always learns the identity of the denoted other endpoint of the channel. In the case of presence of a security annotation symbol, the party learning the identity can be assured of this identity being additionally authenticated.

$$\mathbf{A} \longrightarrow \mathbf{B} \quad (3.1)$$

$$\mathbf{A} \bullet \longrightarrow \mathbf{B} \quad (3.2)$$

$$\mathbf{A} \longrightarrow \bullet \mathbf{B} \quad (3.3)$$

$$\mathbf{A} \bullet \longrightarrow \bullet \mathbf{B} \quad (3.4)$$

When it comes to channel transformations achieved by using cryptographic protocols, the time at which a channel is available is of importance. Thus, the model defines a channel over which a message, fixed or chosen at time t_1 , can be sent at time t_2 to be denoted as $\mathbf{A} \xrightarrow{t_2[t_1]} \mathbf{B}$, where $t_2 > t_1$ must hold. For example, (3.5) shows that a message can only be forwarded from a party **A** to **C** if the time t_3 at which the relaying party **B** can choose its message is after the time t_2 when it has received the original message from party **A**.

$$\mathbf{A} \xrightarrow{t_2[t_1]} \mathbf{B}, \mathbf{B} \xrightarrow{t_4[t_3]} \mathbf{C}, t_3 > t_2 \implies \mathbf{A} \xrightarrow{t_4[t_1]} \mathbf{C} \quad (3.5)$$

Maurer and Schmid conclude that, using the basic cryptographic primitives they discuss, a security property (i.e., a bullet symbol) at one end of a channel existing at time t can be re-established at time t' using an insecure channel given a bullet on the same side of the channel at time t , given $t' > t$. However, they state that two things cannot be achieved through cryptographic protocols: (1) bullets cannot be created, and (2) bullets cannot be moved from one side of the channel to the other.

3.4 Extended Channel Model

We have extended Maurer and Schmid’s model with concepts and transformation rules to additionally accommodate modern cryptographic protocols for obtaining privacy-enhancing authentication and accountability in Bichsel, Camenisch, and Sommer [BCS12a]. This extended model is the basis of the (fragment of) the model presented in the current chapter. Thereby, we also adopt the concept of channels between parties to model security properties between the parties.

We start with presenting the foundations of our model and put it in context with the approach taken by Maurer and Schmid in their work. Like Maurer and Schmid, we use channels to model that parties may exchange information and we use a bullet to annotate a channel endpoint for indicating a security assurance. More concretely, a bullet at the source of a channel denotes an authenticated communication partner at this side of the channel and a bullet at the destination stands for a confidential channel. A channel without bullet annotations does not have any security assurances and is called an insecure channel.

We extend, following the ideas in Bichsel et al. [BCS12a], the Maurer-Schmid approach as follows: (1) parties act under *pseudonyms* instead of under their unique identities, (2) parties are known to other parties through *attribute statements* comprising particularly also their pseudonyms, and (3) the attribute statements about parties are *authenticated* towards the other parties.

3.4.1 Pseudonyms

In the original Maurer–Schmid model, a party is assumed to have a unique, system-wide identifier that identifies the party among all parties in the system and the identifiers are used to refer to the parties in the channel notation. Technology-wise, such an identifier can, e.g., be implemented by the unique public key of the party in a system where each party has exactly one such public key. In each authentic or confidential channel, the party with the security annotation (i.e., the bullet) is known to its communication partner by this unique identifier. This is a core property of the model, based on which channels can be composed to obtain a target channel. A major drawback of this modeling approach is that it cannot reflect the capabilities of today’s privacy-enhanced authentication technologies. We overcome this limitation by allowing parties to have and act under multiple pseudonyms and thereby use a pseudonym as one identifier. Therefore, we define channels to connect two parties known to each other under *pseudonyms* instead of their unique identifiers. Technically, a pseudonym can be seen as the equivalent of a public key in that it (provably) can be related to a secret key. However, a party **A** can generate an arbitrary number of pseudonyms using a single secret key. Note that a party knowing a set of pseudonyms (without the corresponding secret information) cannot distinguish whether or not they have been generated using the same secret key. Thus, pseudonyms are *unlinkable*, that is, cannot be linked to the same party unless further information is available. Furthermore, pseudonyms are *unique*, that is, a pseudonym can only be held by a single party.

Because of the uniqueness of pseudonyms we can define a mapping function $p(\cdot)$ using a pseudonym as input and providing the corresponding party as output. This mapping function p between parties and their pseudonyms is needed for expressing our channel composition rules. More concretely, we use this function to compose channels with different pseudonyms, where the composition requires the party being the holder of those pseudonyms to be the same. Note that this function is not available to parties within the system since this would invalidate the unlinkability property—it is solely used for modeling authentication.

In our channel model we often use an intuitive notation for pseudonyms, where we denote a pseudonym of a party **A** in a communication with party **B** as \mathcal{A}_b . This notational convention should facilitate readability and does not expose any information about the parties. A pseudonym always is a name of a party without carrying further semantics. This notion closely relates to what is denoted as $[A]_i$ by Mödersheim and Vigano [MV09b]. However, our unlinkability property of pseudonyms goes further than their perspective in which pseudonyms model *sender invariance*, where a recipient is assured to be communicating with the same sender (e.g., through the use of an unauthenticated public key). Sender invariance for unauthenticated channels is not explicitly captured in our model, however, it could be achieved, e.g., with using unauthenticated public keys. In any practical system, pseudonyms can be realized through cryptographic mechanisms, for example, using a commitment scheme as in anonymous credential systems [CL01]. A user may generate a polynomial number of pseudonyms \mathcal{A}_i such that uniqueness of the pseudonyms is attained with overwhelming probability. Depending on the cryptographic scheme, the unlinkability property can hold computationally or even information theoretically. We denote a *public pseudonym* of a party **C** as \mathcal{C} , omitting the index, in our notational convention.

In contrast to our earlier channel model [BCS12a], we do not explicitly model the channel endpoints as the pseudonyms of their holding parties. We use the original approach of Maurer and Schmid of denoting channel endpoints with parties and modeling the pseudonyms as part of the statement-based annotation of the channel endpoint and additional predicates. However, we use the terminology of [BCS12a] of a pseudonym being a channel endpoint, with the meaning of the holding party of this pseudonym being the endpoint. The reasons for our modeling choice of not making pseudonyms explicit through the channel endpoints are the following: We align the model with our formal data representation presented in Chapter 4 and not only a less-expressive fragment of it as in our earlier work [BCS12a]. That is, a part of the semantics that has been made explicit in our earlier channel model [BCS12a] is now expressed through the statement annotations and not made explicit in the model. Furthermore, in our generalized approach, a statement annotation ϕ may make statements about multiple parties which cannot be captured using the modeling of pseudonyms as done previously [BCS12a]. As a consequence, pseudonyms and conditional release are not explicitly expressed in our channel model, thereby not having channel conditions expressed on channels and pseudonyms as channel endpoints as Bichsel et al. [BCS12a] do.

3.4.2 Statement-based Party Annotations

For modeling privacy-friendly authentication we not only need to model pseudonyms of parties, but also the exchange of attribute statements about parties. This goes well beyond what Maurer and Schmid can express in their model where authentication is a binary property indicated through the bullet annotation and the precise authentication information remains implicit. Due to the importance of attribute-based authentication in today's information systems and application scenarios, we make an extension to the Maurer–Schmid model to capture this concept.

We implement this by annotating a channel endpoint party with a formula ϕ that expresses the attribute statements associated with the party and that is optionally authenticated by the party. We refer to such annotation as *party annotation*, *endpoint annotation*, or *annotation*, possibly with the prefix “statement-based” to further emphasize the nature of the annotation.

Definition 3.1 (Statement-based annotation) *A statement-based annotation of a party \mathbf{A} being a channel endpoint is defined as the statement(s) ϕ about \mathbf{A} and optionally other parties learnt by the other channel endpoint \mathbf{B} .*

Definition 3.1 means that the channel endpoint party \mathbf{B} learns attribute statements as specified by ϕ , which are expressed as a logic-based formula. In case the endpoint is annotated with a bullet security annotation, this allows its communication partner to derive that the statements are authenticated w.r.t. the semantics of the formula ϕ . Without a bullet, the statement is purely a declaration about an (unverified) statement. Consequently, the presence or absence of a bullet annotation of a party having an attribute-based statement annotation plays the crucial role of assuring the other party whether it communicates with an authenticated party or not. Note that we often use the notion of a party being authenticated w.r.t. a statement synonymously to the more general notion of a party authenticating a statement. Also note that the direction of the channel between \mathcal{A}_b and \mathcal{B}_a is orthogonal to the attribute-based statement annotation and indicates the direction in which a message can be sent over the channel. That is, we assume that the attributes can be learnt by the communication partner even if the channel direction does not suggest so.

A logic-based language for the representation of data statements is introduced in Sections 4.4 and 4.5 and a semantics for this language formally specifying the denotation of statements expressed in the language is presented in Sec. 4.11 of Chapter 4.

3.4.3 Authenticating a Statement

The main process behind our model is the authentication of a statement about parties as captured in Def. 3.2. This serves as a basis for the authentication and confidentiality of a channel that we define later.

Definition 3.2 (Statement authentication) *Statement authentication of a party \mathbf{A} , the authenticating party, is the process of evidence for the truth of a*

data statement (formula) ϕ according to its underlying semantics being provided to and verified by a party \mathbf{B} , the verifier. The formula ϕ can make assertions about \mathbf{A} and possibly other parties $\mathbf{A}_1, \dots, \mathbf{A}_k$, including assertions about pseudonymous identifiers of the parties $\mathbf{A}, \mathbf{A}_1, \dots, \mathbf{A}_k$. The parties $\mathbf{A}, \mathbf{A}_1, \dots, \mathbf{A}_k$ about which statements are made through the formula are denoted as authenticates.

The formula ϕ comprises assertions about authenticates $\mathbf{A}, \mathbf{A}_1, \dots, \mathbf{A}_k$, referred to by pseudonyms, made by or on behalf of \mathbf{A} , and also information about the parties $\mathbf{C}_1, \dots, \mathbf{C}_l$ that certify, that is, vouch for the truth of the assertions, where each \mathbf{C}_i can certify assertions made about any of the authenticates. Parties that certify assertions about other parties and thus vouch for those towards verifiers are denoted as *certifiers*.

The formula ϕ need not identify any of the parties it makes assertions about, and thus parties can remain pseudonymous and known to \mathbf{B} only with respect to a pseudonym and attribute information. This means that authentication is considered in a very broad sense, and releasing attributes about the civil identity of a party is only one very special case. The formula ϕ makes clear which of its assertions are about the authenticating party and other parties by referring to those parties with (pseudonymous) identifiers. The assertion can make very general statements about parties, including conditionally releasing attribute data to third parties. For details on the statement language we refer to the definition of the language in Sections 4.4 and 4.5.

An authentication as explained establishes statements about the authenticating party and possibly further parties towards the verifier \mathbf{B} which can be verified with sufficient strength by \mathbf{B} through trust relations in place between it and certifying parties. The authenticity is given with respect to statements the certifying parties make about \mathbf{A} and possibly other parties which are conveyed to \mathbf{B} .

The term “party” in Def. 3.2 can refer to any entity as in the definition of entity authentication in [MvOV01], e.g., also devices or services. The term “entity” could be used to underline such more generic meaning. We use the term “party” in this work to emphasize that people are our main focus of the model and system proposed in this thesis. The wording “being provided” in Definition 3.2 does not only capture the case where a party \mathbf{A} provides evidence for the truth of a formula, but also third parties can do so. This is, for example, required, when \mathbf{A} uses an authentication mechanism where a third party provides evidence of the truth of the formula on behalf of \mathbf{A} , which is, for example, the case for protocols with online certifiers which authenticate a data statement on behalf of a user.

The definition also captures the case of attribute statements only being made about other authenticates than \mathbf{A} , with only a pseudonym statement being made about \mathbf{A} . The minimal statement made about the authenticating party is a statement about the pseudonym it acts under which is often required for expressing channel composition rules.

Using statement authentication, an authenticated channel between the authenticating party and the verifier can be established. A message originating at the authenticating party is authenticated with the formula ϕ established in the state-

ment authentication for the channel.

The authenticating party is also denoted as *data provider* in this work because it communicates the formula and then performs authentication of it. The authenticator is also denoted as *data recipient* as this captures the role of receiving authenticated data.

Relation to traditional definitions. An important generalization we make in Def. 3.2 in terms of privacy protection over standard definitions of authentication consists in the formula ϕ comprising statements with attribute assertions about the parties that may, but do not necessarily, identify the parties. Particularly, a formula can contain pseudonyms of parties being the subjects of the assertions being made and can express predicates over attributes without revealing the attribute values. This is in stark contrast to the traditional definitions, e.g., the one by Menezes et al. [MvOV01], that assume authenticated parties to be known under a (unique) identifier. Our approach of pseudonym and attribute assertions being made about parties is a major facilitator of privacy due to the data-minimizing statements that can be expressed.

Another notable generalization over traditional definitions is that a formula ϕ used in a statement authentication can express attribute assertions about *multiple parties*. Making authenticated statements about other parties in addition to the authenticating party is required, for example, in the context of the delegation of authority over identity data. This is also a generalization over our definition of *pseudonym authentication*—a form of privacy-enhanced authentication—in our earlier work [BCS12a], which builds on a similar data representation as we do in this thesis, however, it neither makes statements about multiple parties explicit, nor supports them explicitly through channel composition rules.

Also, our definition is not phrased such that the authenticating party authenticates itself, rather the party authenticates a data statement that is about itself and possibly also other parties. Through authenticating the statement, it also authenticates itself w.r.t. the assertions of the statement applying to it. Our definition can be reduced to a special case with the statement ϕ only making assertions about the authenticating party **A**, thereby being congruent to the standard definition for *attribute-based authentication* of a party. By not necessarily having **A** provide the statement to **B**, the definition is also suitable for bootstrapping a system through covering authentications of the party through out-of-band means.

Considering the above, our definition of authentication is substantially more general than the traditional definitions that comprise conveying of identifying information applying to a single entity (party) to the verifier, as, e.g., the definition of entity authentication by Menezes et al. [MvOV01].

3.4.4 Authentication and Confidentiality

As we specify channels between parties acting under pseudonyms, and not under their unique identifiers, we need to update the definition of authentication of channels to fit our model.

Definition 3.3 (Statement authenticated) *A channel from an entity \mathbf{A} with statement annotation ϕ_A and pseudonym \mathcal{A}_b to an entity \mathbf{B} with statement annotation ϕ_B and pseudonym \mathcal{B}_a is statement authenticated if \mathbf{B} is assured of only a party being able to authenticate the statement ϕ_A having access to the other endpoint of the channel and therefore being able to send messages on the channel.*

Definition 3.3 covers the prominent case of ϕ_A making assertions about \mathbf{A} , authenticated by or on behalf of \mathbf{A} . This implies \mathbf{A} being authenticated towards \mathbf{B} through an attribute statement which is a main use case for our secure channel model.

The intuition behind this definition is aligned with the original model, with the difference that \mathbf{B} is assured that it communicates with a party that has authenticated ϕ_A and acts under \mathcal{A}_b instead of being assured that it communicates with party \mathbf{A} that it knows under its unique identifier. The difference articulates in the situation where a party \mathbf{A} repeatedly communicates with another entity. In such case, we can see that \mathbf{A} using the different formulae ϕ_A and ϕ'_A and pseudonyms \mathcal{A}_b and $\mathcal{A}_{b'}$ allows \mathbf{A} to maintain two authenticated but unlinkable channels with her communication partner. Consequently, parties are only linkable when using the same pseudonym or other identifying attributes within the statements on several channels. The definition generalizes the authentication definition in Bichsel et al. [BCS12a] where a—possibly annotated—pseudonym is authenticated.

In the Maurer–Schmid model, the dual property to authentication is *confidentiality*. Dual to the authenticity concept, we introduce the notion of statement confidentiality.

Definition 3.4 (Statement confidential) *A channel from an entity \mathbf{A} with statement annotation ϕ_A and pseudonym \mathcal{A}_b to an entity \mathbf{B} with statement annotation ϕ_B and pseudonym \mathcal{B}_a is statement confidential if \mathbf{A} is assured of only a party being able to authenticate the statement ϕ_B having access to the other endpoint of the channel and therefore being able to receive messages on the channel.*

Clearly, authentication and confidentiality as modeled by Maurer and Schmid are a special case of our extended model, because in their model every party is constrained to having one unique, system-wide identifier. We can obtain their setup as a special case by allowing each entity to use only one pseudonym which represents its unique identity and not allow for general attribute statements about parties. How our changes affect the model can be most easily expressed through the basic channels, i.e., the *insecure*, *authenticated*, *confidential*, and *secure* channel. Next, we present those channels and thereby introduce our notation for expressing our channel model.

Combining both definitions, one can informally characterize a security annotation of a channel endpoint such that presence of a bullet symbol means that the channel endpoint party can authenticate its annotation statement ϕ and has exclusive access to the channel at its side. Note that, even though multiple parties may be able to authenticate a certain attribute statement, only one party can authenticate a specific statement comprising a pseudonym.

Insecure channel. We start with an insecure channel from \mathbf{A} annotated with formula ϕ_A to \mathbf{B} annotated with formula ϕ_B . We model this similarly to the Maurer–

Schmid model, with the difference that parties act and are known to each other under pseudonyms and further attribute statements. Thus, we denote such insecure channel as

$$\mathbf{A}^{\phi_A} \xrightarrow{t_1[t_2]} \phi_B \mathbf{B} . \quad (3.6)$$

In this channel notation, the pseudonyms of the parties are modeled as part of the endpoint annotation formulae, ϕ_A and ϕ_B in the channel in (3.6). Unlike in the Maurer–Schmid model, the channel endpoint parties do not learn the identities of the respective other endpoint in our model. A party only learns the annotation formula, comprising also the pseudonym, of the other endpoint and whether the formula is authenticated.

For making the pseudonyms explicit to express that \mathbf{A} is acting under pseudonym \mathcal{A}_b and party \mathbf{B} under pseudonym \mathcal{B}_a , we can add notational elements and express this as

$$\mathbf{A}^{\phi_A} \xrightarrow{t_1[t_2]} \phi_B \mathbf{B}, \lambda(\phi_A) = \mathcal{A}_b, \lambda(\phi_B) = \mathcal{B}_a . \quad (3.7)$$

The function $\lambda(\phi)$ extracts the pseudonym of the party annotated with the formula ϕ from the formula. In the above example, $\lambda(\phi_A) = \mathcal{A}_b$ and thus also $\mathbf{A} = p(\mathcal{A}_b)$ holds as \mathbf{A} is the annotated party. Making pseudonyms explicit is done implicitly in our earlier model [BCS12a] as the channel endpoints are denoted by the pseudonyms instead of the parties and thus equalities of parties holding pseudonyms need to be stated explicitly. The two modeling approaches are very close to each other and one can claim the differences are mainly of syntactic nature. Making pseudonyms explicit in the current model is required to name the parties towards each other when applying a sequence of transformation rules for composing channels. Rules are only applicable in case also the pseudonyms of parties match.

Note that, because the channel in (3.1) is insecure, the index of a pseudonym denotes the *intended* communication partner, e.g., \mathcal{A}_b for \mathbf{A} communicating with \mathbf{B} .

We can look at an insecure channel in two different ways. First, it visualizes the security information (authentication or confidentiality) available to the communicating parties. From this point of view, the party $\mathbf{A} = p(\mathcal{A}_b)$ may be any party in the system. This results from the fact that the pseudonym does not have a security annotation (i.e., a bullet symbol at the endpoint). \mathcal{A}_b here is simply a name used to refer to the pseudonym of the *intended channel endpoint*. Party $\mathbf{B} = p(\mathcal{B}_a)$ learns only the unauthenticated formula ϕ_A including a pseudonym about its communication partner. This is what we define as an insecure channel: similar to using an unauthenticated public key, the pseudonym does not imply communication with the party legitimately holding the pseudonym. Second, an insecure channel denotes the availability of a channel between the parties. For our channel transformations presented later we often use insecure channels between two parties to denote that the parties have access to a communication channel.

Authentic channel. An example of a channel from \mathbf{A} , authenticating and thus authenticated through ϕ_A and pseudonym \mathcal{A}_b , to \mathbf{B} , known to \mathbf{A} under unauthenticated formula ϕ_B and acting under the unauthenticated pseudonym \mathcal{B}_a is denoted

as

$$\mathbf{A}^{\phi_A} \xrightarrow{t_2[t_1]} \phi_B \mathbf{B}, \lambda(\phi_A) = \mathcal{A}_b, \lambda(\phi_B) = \mathcal{B}_a. \quad (3.8)$$

Note that \mathbf{B} only knows the other channel endpoint as specified through the formula ϕ_A , particularly as holding pseudonym \mathcal{A}_b . This results from the unlinkability of pseudonyms as well as the fact that parties within the system do not have access to the function p to map pseudonyms to parties. Party \mathbf{A} can send messages authenticated with ϕ_A to \mathbf{B} over this channel where the former does not have any (authentic) information on the statement ϕ_B including pseudonym \mathcal{B}_a about the latter. This is the notation of a statement-authenticated channel based on the notation of an authenticated channel in the model of Maurer and Schmid where authentication is defined in a more restrictive way through a party authenticating under its system-wide identifier.

A statement-authenticated channel from \mathbf{A} to \mathbf{B} with statement ϕ implies that \mathbf{B} can be ensured that only a party which is able to establish the validity of ϕ —or can have it established—can send a message to \mathbf{B} over this channel. Technically, \mathbf{A} itself or another party on behalf of it can perform the authentication of the statement and establish this validity, without this resulting in a difference in authentication semantics, however, possibly resulting in a difference of the obtained privacy properties of the authentication protocol.

Confidential channel. A confidential channel is generalized similar to an authentic channel. Instead of knowing that the channel is established with an entity specified by a unique identifier, a message sent over a statement-confidential channel is known to be exclusively available to a party being able to authenticate its annotation statement, ϕ_B in the example channel (3.9). Particularly, the recipient is also specified through ϕ_B , including the pseudonym it acts under. In an example, we denote a pseudonym-confidential channel from a pseudonymous party \mathbf{A} acting under \mathcal{A}_b to a party \mathbf{B} acting under \mathcal{B}_a as

$$\mathbf{A}^{\phi_A} \xrightarrow{t_2[t_1]} \phi_B \mathbf{B}, \lambda(\phi_A) = \mathcal{A}_b, \lambda(\phi_B) = \mathcal{B}_a. \quad (3.9)$$

In this example, \mathbf{A} can be assured that only the party being able to authenticate a formula ϕ_B , including pseudonym \mathcal{B}_a , has access to the channel, that is, can read messages sent over the channel.

Secure channel. A secure channel between parties with annotations ϕ_A and ϕ_B , acting under pseudonyms \mathcal{A}_b and \mathcal{B}_a , assures the parties $\mathbf{A} = p(\mathcal{A}_b)$ and $\mathbf{B} = p(\mathcal{B}_a)$, holding the pseudonyms \mathcal{A}_b and \mathcal{B}_a , that their communication partner is the party being able to authenticate the formula ϕ_A and ϕ_B , respectively. We denote a secure channel as

$$\mathbf{A}^{\phi_A} \xrightarrow{t_2[t_1]} \phi_B \mathbf{B}, \lambda(\phi_A) = \mathcal{A}_b, \lambda(\phi_B) = \mathcal{B}_a. \quad (3.10)$$

Note that we simplify the notation in the remainder of the paper by using the phrase that a pseudonym \mathcal{A}_b has a channel to a pseudonym \mathcal{B}_a as shorthand for the party $\mathbf{A} = p(\mathcal{A}_b)$, i.e., party \mathbf{A} holding pseudonym \mathcal{A}_b , having a channel to party $\mathbf{B} = p(\mathcal{B}_a)$.

3.5 Channel Transformation Algebra

For defining a channel transformation algebra, we need to specify the transformation rules for our extended channel model. The transformation rules of the original model of Maurer and Schmid can be carried over to our extended model by enriching them with our extensions. We refer to those transformations as *basic channel transformations* and provide some examples of how we change the original rules to the setting of our model, without doing this exhaustively for all rules. Additionally, we require new rules for specifying how party annotations and pseudonyms of parties are handled in channel transformations. Those new rules are referred to as *statement-based transformations* in this thesis.

The rule set comprising the basic channel transformation rules and the statement-based transformation rules as presented in this section form the basis for our extended Maurer–Schmid channel composition algebra. This small rule set is sufficient for the channel derivation calculus we propose.

There are three ways how an authentication relationship in the form of an authenticated channel from party **A** to a party **C** can be established: (1) through out-of-band authentication, (2) through party **C** assigning attributes to **A**, or (3) through cryptographic transformations based on already-existing authentic channels using channel composition rules. In terms of our symbolic notation, option (1) and (2) are reflected through existing authentic channels from **A** to **C** with the security annotation on the side of **A**. Option (3) reflects applications of rules of our channel composition algebra to obtain new channels from existing channels. There is no difference in terms of the security annotation of a statement depending on the way a channel is established. The different ways of establishing an authentication can also be combined to obtain a single channel with authenticated endpoint annotation.

Insecure channels exist between parties in an open communication infrastructure such as the Internet at certain times and represent the possibility of parties to communicate with each other. Some initial secure channels between parties need to be assumed to exist, established through option (1) and optionally also (2) as explained above in an initial setup phase of a system.

Based on the available secure and insecure channels between parties, new secure channels with desired security—particularly authentication—properties can be constructed using cryptographic transformations. Maurer and Schmid’s model and algebra are based on traditional cryptographic mechanisms such as symmetric cryptography, message authentication codes (MACs), public key encryption schemes, and signature schemes. We next present our extension of Maurer and Schmid’s algebra towards privacy-enhanced cryptographic credential protocols. When applying a channel transformation rule, bullet annotations on prerequisite channels of the rule may—as in Maurer and Schmid’s model—be dropped as this only results in a weaker channel, derivability of which is implied by the one with annotation.

We need to use the function $\lambda(\phi)$ on a statement ϕ introduced earlier for specifying relevant aspects of the transformation rules: Through the function λ we make pseudonyms of parties explicit as part of a transformation rule and relate pseudonyms of different formulae with each other. Further notational elements re-

quired for some rules will be introduced with those rules for ease of reading.

3.5.1 Basic Transformations

The channel transformation rules of Maurer and Schmid need to be adapted to fit our extended model. We show this through some example rules to present the idea how this is done, however, we do not exhaustively extend all basic rules.

3.5.1.1 Symmetric Cryptography

The rule for transferring a secure channel using an insecure channel available at a later time extends to Rule **Sec-Symm** in our model, presented in (3.11) below.

$$\begin{aligned}
 &\text{Sec-Symm} \\
 &\mathbf{A}^{\phi_{A,1}} \bullet \xrightarrow{t_2[t_1]} \phi_{B,1} \mathbf{B}, \mathbf{A}^{\phi_{A,2}} \xrightarrow{t_4[t_3]} \phi_{B,2} \mathbf{B} \\
 &\lambda(\phi_{A,1}) = \lambda(\phi_{A,2}) = \mathcal{A}_b, \lambda(\phi_{B,1}) = \lambda(\phi_{B,2}) = \mathcal{B}_a \\
 &\implies \mathbf{A}^{\phi_{A,1}} \bullet \xrightarrow{t_4[t_3]} \phi_{B,1} \mathbf{B}
 \end{aligned} \tag{3.11}$$

This rule transfers the security properties of a channel to an insecure channel available at a later point in time using symmetric cryptography. Concretely, this transformation rule can be realized with a mutually authenticated shared secret key. In addition to Maurer and Schmid's rule we build upon, the rule features party annotations, that, among others, specify the pseudonyms parties act under.

3.5.1.2 Public-key Encryption

The transformation rule **AuthCon-PKEnc** in (3.12) specifies that an authenticated channel from \mathbf{A} annotated with $\phi_{A,1}$ and acting under \mathcal{A}_b to \mathbf{B} annotated with $\phi_{B,1}$ and acting under \mathcal{B}_a over which a message fixed at time t_1 can be sent at time t_2 can be used to create a confidential channel from \mathcal{B}_a to \mathcal{A}_b at a later time.

$$\begin{aligned}
 &\text{AuthCon-PKEnc} \\
 &\mathbf{A}^{\phi_{A,1}} \bullet \xrightarrow{t_2[t_1]} \phi_{B,1} \mathbf{B}, \mathbf{A}^{\phi_{A,2}} \xleftarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}, \\
 &\lambda(\phi_{A,1}) = \lambda(\phi_{A,2}) = \mathcal{A}_b, \lambda(\phi_{B,1}) = \lambda(\phi_{B,2}) = \mathcal{B}_a, t_3 > t_2 \\
 &\implies \mathbf{A}^{\phi_{A,1}} \bullet \xleftarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}
 \end{aligned} \tag{3.12}$$

This can be easily achieved by \mathbf{A} authenticating a public encryption key for which it holds the private key as message over the first channel. In the setting of credential systems, the authentication of the encryption key can be achieved through a Fiat-Shamir signature [FS86] based on the zero-knowledge proof of knowledge. This amounts to applying the concept of *credential-authenticated key exchange* for the public key.

3.5.1.3 Transferring Authenticity

Rule **Auth-Sig** presented in (3.13) is the extension of the rule for transferring the authenticity property of a channel between party **A** annotated with statement $\phi_{A,1}$ to party **B** annotated with $\phi_{B,1}$ to an insecure channel available at a later point in time between the parties to fit our extended model.

$$\begin{aligned}
 &\text{Auth-Sig} \\
 &\mathbf{A}^{\phi_{A,1}} \bullet \xrightarrow{t_2[t_1]} \phi_{B,1} \mathbf{B}, \mathbf{A}^{\phi_{A,2}} \xrightarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}, \\
 &\lambda(\phi_{A,1}) = \lambda(\phi_{A,2}) = \mathcal{A}_b, \lambda(\phi_{B,1}) = \lambda(\phi_{B,2}) = \mathcal{B}_a, t_3 > t_2 \\
 &\implies \mathbf{A}^{\phi_{A,1}} \bullet \xrightarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}
 \end{aligned} \tag{3.13}$$

Such transfer of authenticity without secrecy is a transformation that can be realized with a signature scheme, while transferring both authenticity and secrecy of a channel can be achieved with symmetric cryptography using a shared symmetric key as discussed earlier [MS94].

Once the authentication of $\phi_{A,1}$ has been obtained through a private certificate protocol, the target channel of the rule can be obtained through a signature of the message to be sent over this channel using the pseudonym \mathcal{A}_b under which **A** acts on the first channel.

3.5.1.4 Trust-based Transformation

The generalization of the rule of the Maurer–Schmid model for transferring authenticity to a later channel based on trust to fit our extensions is presented next. Rule **ConnectAnn-Trust** in (3.14) transfers the authenticity property for the annotation ϕ_1 of a channel between **A** and **T** to a channel from party **A** to **B**, using a channel from **T** to **B**. The rule can be viewed as an extension of Rule **ConnectAnn** with **A** authenticating a statement ϕ_A to **T** instead of the statement being non-authenticated. The rule models the trust-based transformation of the security properties of the first channel to the target channel $\mathbf{A}^{\phi_1} \bullet \xrightarrow{t_4[t_1]} \phi_B \mathbf{B}$.

$$\begin{aligned}
 &\text{ConnectAnn-Trust} \\
 &\mathbf{A}^{\phi_1} \bullet \xrightarrow{t_2[t_1]} \phi_{T,A} \mathbf{T}, \mathbf{T}^{\phi_{T,B}} \bullet \xrightarrow{t_4[t_3]} \phi_B \mathbf{B}, \\
 &\lambda(\phi_1) = \mathcal{A}_b, \lambda(\phi_{T,A}) = \mathcal{T}_a, \lambda(\phi_{T,B}) = \mathcal{T}_b, \lambda(\phi_B) = \mathcal{B}, \\
 &\mathbf{B} \text{ trusts } \mathbf{T}, t_3 > t_2 \\
 &\implies \mathbf{A}^{\phi_1} \bullet \xrightarrow{t_4[t_1]} \phi_B \mathbf{B}
 \end{aligned} \tag{3.14}$$

The rule shows how a party annotation ϕ_1 of **A** can be transferred from the first channel to the target channel, using a party **T** as trusted intermediary. That is, it

models the transfer of ϕ_1 in an integrity-protected way to **B** through **T**, for which trust of the recipient in **T** is required.

A noteworthy aspect of this rule is that the second prerequisite channel of the rule needs a security annotation on the side of **T** because otherwise the required trust relation does not have any meaning as **T** could be any party. **B** does not need to be authenticated to **T**—in case a security annotation for it is required on the target channel, the rule needs to be composed with another rule for obtaining the annotation.

The target channel does not require direct communication, that is, an insecure channel, between **A** and **B** being available. Still, **A** also learns ϕ_B , as specified through the target channel of the rule, for the following reason: This rule models that **A** requests from **T** that **T** forward the message to a party that is known under statement ϕ_B and pseudonym $\lambda(\phi_B) = \mathcal{B}$.

Note that **T** can act under different pseudonyms and have different annotations towards parties **A** and **B**. This generalizes the model of Bichsel et al. [BCS12a] which constrains **T** to act under a public pseudonym. Party **B** always acts under the same (public) pseudonym towards both parties which simplifies handling of the related annotation formula ϕ_B .

3.5.2 Statement-based Transformations

The new transformation rules required for modeling channel transformations considering statement-based party annotations and pseudonyms and defining how statement-based party annotations are propagated between channels are presented next. Both rules that operate on unauthenticated as well as authenticated annotation statements are considered. The rules with authenticated annotation statements are most interesting to us as they model the cryptographic protocols discussed later in this thesis.

It is, to the best of our knowledge, not possible to transfer the authentications orthogonally to the statement-based endpoint annotations in all situations. Thus, Rule **CombineAnn** for combining statement annotations is of limited practical applicability because in order to obtain a target channel with authentication of ϕ_3 using orthogonal authentication transfer requires transfer of authentication for the combined formula ϕ_3 . In practice, we are mainly interested in combining channels with authenticated annotations as in the rule for combining authenticated statement annotations modeling credential protocols.

3.5.2.1 Combining Statement Annotations

A basic rule **CombineAnn**, presented in (3.15) below, defines how pseudonym annotations of two channels between the same entities can be combined. This is relevant in a scenario where two parties **A** and **B** repeatedly communicate using the same

pseudonyms \mathcal{A}_b and \mathcal{B}_a .

CombineAnn

$$\begin{aligned} & \mathbf{A}^{\phi_1} \xrightarrow{t_2[t_1]} \phi_{B,1} \mathbf{B}, \mathbf{A}^{\phi_2} \xrightarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}, \\ & \lambda(\phi_1) = \lambda(\phi_2) = \mathcal{A}_b, \lambda(\phi_{B,1}) = \lambda(\phi_{B,2}) = \mathcal{B}_a \\ & \implies \mathbf{A}^{\phi_3} \xrightarrow{t_4[t_3]} \phi_{B,2} \mathbf{B}, \phi_3 = \phi_1 \wedge \phi_2, \lambda(\phi_3) = \lambda(\phi_1) \end{aligned} \quad (3.15)$$

The intuition behind this rule is that statements made by a party \mathbf{A} acting under a pseudonym with a party \mathbf{B} over different channels can be combined into a new channel revealing a statement that is the conjunction of both statements.

For this and selected other rules, only the pseudonym of the annotation function of \mathbf{B} is relevant. A conjunction of the formulae \mathbf{B} acts under can, in the case of this being interesting in a given situation, be done by application of an appropriate transformation rule.

3.5.2.2 Combining Authenticated Statement Annotations

Combining authenticated statement-based annotations is the generalization of the rule for combining statement-based party annotations to authenticated statement-based party annotations. The rule **CombineAuthAnn-Cred** in (3.16) specifies how the authenticated annotations of a party \mathbf{A} of two channels with the same other party \mathbf{B} can be combined to a single annotation on a new channel when the parties use the same pseudonyms.

CombineAuthAnn-Cred

$$\begin{aligned} & \mathbf{A}^{\phi_1} \xrightarrow{\bullet, t_2[t_1]} \phi_{B,1} \mathbf{B}, \mathbf{A}^{\phi_2} \xrightarrow{\bullet, t_4[t_3]} \phi_{B,2} \mathbf{B}, \\ & \lambda(\phi_1) = \lambda(\phi_2) = \mathcal{A}_b, \lambda(\phi_{B,1}) = \lambda(\phi_{B,2}) = \mathcal{B}_a \\ & \implies \mathbf{A}^{\phi_3} \xrightarrow{\bullet, t_4[t_3]} \phi_{B,2} \mathbf{B}, \phi_3 = \phi_1 \wedge \phi_2, \lambda(\phi_3) = \lambda(\phi_1) \end{aligned} \quad (3.16)$$

To the best of our knowledge, no cryptographic protocols that would combine ϕ_1 and ϕ_2 with other operations than conjunction exist. This rule can, like any other rule, be applied recursively to combine security properties from $k > 2$ channels into a single newly created channel.

The rule models multiple pseudonymous statement authentications \mathbf{A} has with \mathbf{B} under the same pseudonym. Cryptographically, this can be realized with multiple instances of authentication protocols using private certificates and exposing the same pseudonym of \mathbf{A} . Note that this also captures the case of \mathbf{A} making multiple authenticated statements about a delegator \mathbf{D} to \mathbf{B} . Both the pseudonyms for \mathbf{A} and \mathbf{D} are the same in both prerequisite channels of the rule in this case. The rule models only the combination of annotations of \mathbf{A} and if one is interested in the annotations of party \mathbf{B} , this needs to be handled independently.

3.5.2.3 Reducing a Statement-based Annotation

Rule **ReduceAnn** in (3.17) expresses that a channel where **A** is known under a formula ϕ_A by a party **B** also represents a channel where **A** is known by a formula ϕ'_A that follows syntactically from ϕ_A in the underlying calculus.

$$\begin{aligned}
 &\text{ReduceAnn} \\
 &\mathbf{A}^{\phi_A} \xrightarrow{t_2[t_1]} \phi_B \mathbf{B}, \lambda(\phi_A) = \mathcal{A}_b, \lambda(\phi_B) = \mathcal{B}_a \\
 &\implies \mathbf{A}^{\phi'_A} \xrightarrow{t_2[t_1]} \phi_B \mathbf{B}, \phi_A \vdash_{\mathcal{O}} \phi'_A
 \end{aligned} \tag{3.17}$$

We do not formalize the corresponding rule for a security annotation being present on the side of **A** here, because cryptographically an existing authentication based on credential protocols cannot be “reduced” like this. However, the general concept that, when a statement ϕ_A is authenticated towards a party, then also a formula ϕ'_A , where $\phi_A \vdash_{\mathcal{O}} \phi'_A$, is authenticated, clearly applies, where \mathcal{O} is an ontology as introduced later in this work.

3.5.2.4 Connecting Statement-annotated Channels

Another new rule, **ConnectAnn**, specifies how annotations propagate to a new channel that is created through a party **T** connecting two channels. Similar to the trust-based rule **ConnectAnn-Trust** that allows for propagation of a security annotation, trust enables propagation of statement-based endpoint annotations.

$$\begin{aligned}
 &\text{ConnectAnn} \\
 &\mathbf{A}^{\phi_1} \xrightarrow{t_2[t_1]} \phi_{T,A} \mathbf{T}, \mathbf{T}^{\phi_{T,B}} \bullet \xrightarrow{t_4[t_3]} \phi_B \mathbf{B}, \\
 &\lambda(\phi_1) = \mathcal{A}_b, \lambda(\phi_{T,A}) = \mathcal{T}_a, \lambda(\phi_{T,B}) = \mathcal{T}_b, \lambda(\phi_B) = \mathcal{B}, \\
 &\mathbf{B} \text{ trusts } \mathbf{T}, t_3 > t_2 \\
 &\implies \mathbf{A}^{\phi_1} \xrightarrow{t_4[t_1]} \phi_B \mathbf{B}
 \end{aligned} \tag{3.18}$$

The discussions for Rule **ConnectAnn-Trust** carry over to this rule and apply analogously.

3.5.2.5 Transfer of Authentications with k Certifiers

The use of a credential system for authentication is modeled with Rule **Auth-Cred**. This rule models that party **A** obtains credentials from the certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$ and uses the credentials for authenticating an annotation statement ϕ' to another entity, **B**.

Auth-Cred

$$\begin{aligned}
& \left(\mathbf{A}^{\phi_i} \bullet \xrightarrow{t_{i2}[t_{i1}]} \phi_{C_i,A} \mathbf{C}_i, \mathbf{C}_i^{\phi_{C_i,B}} \bullet \xrightarrow{t_{i4}[t_{i3}]} \phi_{B,C_i} \mathbf{B}, \right. \\
& t_5 > t_{i2}, t_5 > t_{i4}, \mathbf{B} \text{ trusts } \mathbf{C}_i, \\
& \left. \lambda(\phi_i) = \mathcal{A}_{C_i}, \lambda(\phi_{C_i,A}) = \mathcal{C}_{i,a}, \lambda(\phi_{C_i,B}) = \mathcal{C}_{i,b}, \lambda(\phi_{B,C_i}) = \mathcal{B}_{C_i}, \right)_{\forall 1 \leq i \leq k} \quad (3.19) \\
& \mathbf{A}^{\phi'} \xrightarrow{t_6[t_5]} \phi_{B,A} \mathbf{B}, \lambda(\phi') = \mathcal{A}_b, \lambda(\phi_{B,A}) = \mathcal{B}_a, \\
& \mathcal{F}^P \left(\left(\bigwedge_{i=1}^k \mathcal{F}^R(\mathcal{F}(\phi_i, \mathbf{C}_i, \mathbb{N}_i), \mathcal{P}_{\mathbf{A}}) \right) \wedge \theta_{\mathbf{A},\mathbf{B}}^{t_5} \wedge \eta_{\mathbf{A},\mathbf{B}}^{t_5} \wedge \chi_{\mathbf{A},\mathbf{B}}^{t_5} \right) \vdash_{\mathcal{O}} \phi' \\
& \implies \mathbf{A}^{\phi'} \bullet \xrightarrow{t_6[t_5]} \phi_{B,A} \mathbf{B}
\end{aligned}$$

The rule requires as prerequisite that party \mathbf{A} have authentic channels with certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$ at times $t_{i2}, 1 \leq i \leq k$. Those authentic channels model the authentication of a statement of \mathbf{A} with the certifiers based on which the certifiers issue private certificates to \mathbf{A} . Those certificates are used at a later time by \mathbf{A} to authenticate a statement ϕ' to \mathbf{B} . Certificates can, once obtained, be used for multiple authentications to other entities due to the multi-show unlinkability property of credential systems. Expressed in terms of the rule, a prerequisite channel modeling \mathbf{A} obtaining a credential can be reused for multiple applications of the rule.

The timing condition $t_5 > t_{i2}$ models that the statement on the target channel can only be fixed once \mathbf{A} has obtained the credentials from the certifiers because authentication is done through a protocol based on the credentials. A similar argumentation holds for the condition $t_5 > t_{i4}$ relating the time for \mathbf{B} obtaining the public keys of the certifiers and that of the target channel. \mathbf{B} needs to trust the certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$ in the standard notion of Maurer and Schmid which is expressed with the *trusts* predicates in the rule. Furthermore, authentic channels between each certifier and \mathbf{B} need to exist over which \mathbf{B} can obtain their authentic public keys to verify proofs \mathbf{A} makes based on credentials. Those authentic channels can be realized with channel transformation rules modeling standard public key infrastructure (PKI) protocols [MS94, MS96].

The channel $\mathbf{A}^{\phi'} \xrightarrow{t_6[t_5]} \phi_{B,A} \mathbf{B}$ with the unauthenticated annotation ϕ' on \mathbf{A} 's side models the statement ϕ' with which \mathbf{A} is annotated towards \mathbf{B} . The rule transfers the k authentications of \mathbf{A} with annotations ϕ_i with the certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$ to this channel for obtaining the target channel $\mathbf{A}^{\phi'} \bullet \xrightarrow{t_6[t_5]} \phi_{B,A} \mathbf{B}$. Expressed differently, the bullet security symbols of the channels with the certifiers are transferred to a single security symbol for the annotation formula ϕ' on the target channel.

Multiple new notational elements as outlined next are required for expressing the relationship between the formulae ϕ_i for $1 \leq i \leq k$ and the target formula ϕ' . Those elements are closely related to the technical modeling of data as discussed in

Chapter 4. Those have not been required in our earlier model [BCS12a] mainly due to the higher level of abstraction and the different modeling approach.

$\theta_{\mathbf{A},\mathbf{B}}^{t_5}$ comprises the conjunction of all specification formulae of identifier objects that party \mathbf{A} has previously established with \mathbf{B} about itself and other parties, corresponding to pseudonyms. Those need to be expressed separately for the reason that the pseudonym statements are not captured by the formulae ϕ_i . For expressing conditional release, we introduce $\eta_{\mathbf{A},\mathbf{B}}^{t_5}$ which captures sets of conditionally released attributes created by \mathbf{A} which ϕ' can make statements about, again as a conjunction of formulae. Analogously, $\chi_{\mathbf{A},\mathbf{B}}^{t_5}$ specifies opaque identities to be established by \mathbf{A} with \mathbf{B} . This way of modeling of conditionally released identities and opaque identities results from the definition of the semantics of our data representation language, see Chapter 4 for details regarding this. The timestamp t_5 in $\theta_{\mathbf{A},\mathbf{B}}^{t_5}$, $\eta_{\mathbf{A},\mathbf{B}}^{t_5}$, and $\chi_{\mathbf{A},\mathbf{B}}^{t_5}$ indicates that those structures be considered as of this time. The time t_5 specifies when the message on the target channel needs to be fixed. In the earlier model [BCS12a], conditional release has been modeled as the release of a data statement over a channel with which a condition has been associated as a generalized notion of the channel timings of the original Maurer–Schmid model, thus, the model also requires specific notational elements. Those make, however, the conditional release more prominent in terms of the channel model.

The formula ϕ_i under which \mathbf{A} is authenticated towards \mathbf{C}_i needs to undergo a transformation \mathcal{F} by \mathbf{C}_i , and the result of this needs to undergo a transformation \mathcal{F}^R by \mathbf{A} . The result of applying both transformations is a formula that represents the data \mathbf{C}_i vouches for using \mathbf{A} 's data representation. Those transformations account for properties of the data representation of Sections 4.4 and 4.5 and are required for accurately expressing the relations between the parties.

$\mathcal{F}(\phi_i, \mathbf{C}_i, \aleph_i)$ refers to the certifier part of the transformation that a statement ϕ_i under which a party \mathbf{A} is authenticated towards a certifier \mathbf{C}_i needs to undergo in order that it can be used as one of the formulae which the target formula ϕ' is derived from. Thereby, \mathcal{F} models aspects that depend on a variety of factors related to an issuing transaction of a credential and cannot be expressed more explicitly in the channel model. Because this information is required for expressing the target formula ϕ' , it needs to be expressed as an additional notational element in the specification of the transformation rule. The function is parameterized with the certifier \mathbf{C}_i and a transaction context \aleph_i that abstractly captures the full context determining the transformation.

\mathcal{F}^R refers to \mathbf{A} 's part of the transformation to obtain a formula the target formula ϕ' is derived from. As explained in greater detail in Sec. 5.3, this function represents a term renaming on the formula in order to obtain terms used by \mathbf{A} . Note that $\mathcal{P}_{\mathbf{A}}$ refers to party \mathbf{A} 's data repository required for the processing.

Details on the transformations \mathcal{F} and \mathcal{F}^P are presented in Chapter 5—they rely on concepts introduced in Chapter 4 and thus cannot be treated at this point in detail. We anticipate only that \mathcal{F} removes pseudonym statements of its argument ϕ_i , as well as certification metadata and, optionally, also attribute statements. As well, it introduces a new term to refer to the new identity (certificate) being created through the credential issuing.

The target formula ϕ' must be derivable from a conjunction of the formulae $\mathcal{F}^R(\mathcal{F}(\phi_i, \mathbf{C}_i, \mathbb{N}_i), \mathcal{P}_A)$ for $1 \leq i \leq k$ with respect to the $\vdash_{\mathcal{O}}$ -operator of our logic discussed in Sec. 4.9, the pseudonym statements $\theta_{A,B}^{t_5}$ of pseudonyms A has established with B , and the conditionally released attribute statements $\eta_{A,B}^{t_5}$ and opaque attribute statements $\chi_{A,B}^{t_5}$, after applying a term renaming function \mathcal{F}^P that implements a renaming of terms used by A locally to terms exposed to a data recipient.

The decoration \mathcal{O} of the \vdash -symbol in the rule refers to the use of an ontology considered for the deductions. Details of this are presented in Sec. 4.10, for now we anticipate that the ontology comprises a formal specification of identity types for having the necessary information for being able to derive formulae comprising disjunction statements referring to identities (credentials) the party does not have identity relationships (credentials) for. Without the ontology being considered, only simpler, and less data-minimizing, statements would be possible to be derived as no information on credentials not held by A would be available to the party. This limitation holds because of the nature of deductions in first-order logic, or our calculus based on such logic.

Note that because of how the formulae ϕ_i and ϕ' are related through the \vdash -symbol with its standard meaning in first-order logic and our ontology extension, it is not possible to express this rule through a recursive application of a rule with only one certifier because ϕ' can be composed in more general ways than through conjunction of formulae $\mathcal{F}(\phi_i, \mathbf{C}_i, \mathbb{N}_i)$. This more general composition is a substantial advantage of the employed cryptographic protocols which can prove such formulae correct over traditional protocols.

In terms of authentication, the rule transforms k channels with authenticated annotation formulae ϕ_i of a party A with parties \mathbf{C}_i , $1 \leq i \leq k$, authentic channels of a verifier B with the parties \mathbf{C}_i , an insecure channel between A and B , and trust relationships between B and the certifiers to an authenticated channel from A to B where A can authenticate a statement annotation ϕ' based on the formulae ϕ_i , pseudonym statements, and conditionally released and opaque identity specifications in a generic way. Expressing the intuition behind the rule differently, the authentications of A with the parties \mathbf{C}_i are transferred to the target channel of A with B .

Discussion. The authentication transfer expressed in the rule is exactly what is achieved through a proof in zero knowledge of the correctness of a pseudonym and attribute statement ϕ' based on the credentials party A has obtained from $\mathbf{C}_1, \dots, \mathbf{C}_k$. Thus, the credentials are used as cryptographic tool for transferring authentications between parties. This relates closely to Chaum's terminology of transferring credentials between organizations in his less general model of attribute-less credentials.

A different approach to abstractly viewing the certification of data than as a transfer of authentication is the following: a statement about a party A is expressed and vouched for by a certifier C regardless of the statement A is authenticated under towards C . Of course, this does not prevent C from taking such authentications

into account, but does not limit it to those statements. Trust of verifiers in **C** and the statements it makes allows us to establish authentications between **A** and the verifiers based on the statements by **C** about **A**. In this model, only an authentication of **A** towards **C** must be in place such that it can be ensured that the statement of **C** is assigned to the proper party **A**, but the statement is not necessarily the main input determining the statement **C** vouches for.

When viewing a statement that a certifier **C** makes about a party **A** as an authenticated statement annotation of **A** towards **C** in the model, both above views of certification of data are combined in the model.

In a practical setting, a channel with an authenticated annotation of **A** towards **C_i** can be composed from multiple authenticated channels between the parties where **A** acts under the same pseudonym using Rule **CombineAuthAnn-Cred** and particularly captures also the case of **C_i** making additional statements about **A** which are seen as trivially authenticated statements on the endpoint **A**. This allows for always obtaining a single channel of **A** with an authenticated annotation formula with each certifier as required by Rule **Auth-Cred**.

Of particular interest is the relation of the annotation formulae of the channel of **A** with parties **C_i** and the target channel in terms of pseudonyms as expressed in the rule: The party **A** can be known to all parties **C_i** and party **B** under different pseudonyms, namely $\lambda(\phi_i) = \mathcal{A}_{c_i}$ and $\lambda(\phi') = \mathcal{A}_b$ and—as formalized further above—the formula on the target channel can comprise partial information of the formulae ϕ_i .

Note that the authenticity property of a channel between **A** and **B** obtained using Rule **Auth-Cred** and credential systems as underlying technology can be transferred to a channel that is available at a later time between **A** and **B** using rules **Sec-Symm** or **Auth-Sig** and their cryptographic transformations.

Note also that the transformation rule does not consider prerequisite channel requirements for supporting revocation of credentials. Revocation may require additional prerequisite channels to the certifiers or third parties being available at a time before authenticating ϕ' in order to allow **A** to update the credentials it has previously obtained. We do not further elaborate on this and leave those extensions to future work. The concepts available in the model are sufficient for extending our model for reflecting this.

Modeling traditional protocols. Using conventional certificates is captured as well by the rule, however, constrains it in multiple ways, most notably, such that the pseudonym of **A** used with a certifier and the verifier is the same and the information released by **A** as part of ϕ' for a certificate is all information comprised in $\mathcal{F}(\phi_i, \mathbf{C}_i, \mathbf{N}_i)$ and both \mathcal{F}^R and \mathcal{F}^P are the identity function. Structurally, the rule applies to conventional certificate protocols.

The rule does not capture certification protocols with an online certifier that vouches for statements on behalf of party **A** towards **B** instead of **A** executing the protocol with certificates obtained from the certifier. In such protocols, it is possible to reduce the information formula ϕ' comprises compared to a ϕ_i , much like in Rule **Auth-Cred** modeling private certificate protocols. The reason for this

not being captured is the lack of prerequisite channels involving the certifier at the time the formula ϕ' is to be authenticated.

3.5.2.6 Delegation of Attribute Authority

A crucial feature we are interested in expressing in our model is *delegation of authority* over attribute statements as discussed in Sec. 2.8. Such delegation allows a party **D**, the *delegater*, to delegate authority over authenticating a statement—or statements derivable from this statement in a calculus—to a *delegatee* **A**, where **D** is itself capable of authenticating the statement. Using credential protocols, authority over the statements that a credential holder (i.e., the delegater) can make based on a credential—or a subset thereof—may be delegated.

Rule **AuthDele-Cred** presented in (3.20) models delegation in combination with the transfer of k authentications with certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$ of Rule **Auth-Cred**. This combination is required in order to model a general statement ϕ' being made by **A** based on its authentications with certifiers in conjunction with the delegated statement.

The rule has the same prerequisites as Rule **Auth-Cred** regarding authentic channels with certifiers $\mathbf{C}_1, \dots, \mathbf{C}_k$, trust of **B** in those, and authentic channels from the certifiers to **B** for transferring an authentic credential verification public key to **B**. This extends to **B** trusting certifier \mathbf{C}_{k+1} and an authentic channel from \mathbf{C}_{k+1} to **B** being in place for allowing the latter to also obtain the authentic public credential verification key of \mathbf{C}_{k+1} . For further details on those prerequisite channels, see the discussions related to Rule **Auth-Cred**.

Delegation establishment. The delegation establishment is modeled as follows: The delegater **D** has a channel with an authenticated annotation ϕ_D to certifier \mathbf{C}_{k+1} for obtaining a credential that is to be delegated later. The delegation is initiated by **D** through authenticating an annotation statement ϕ_D^* towards \mathbf{C}_{k+1} with $\lambda(\phi_D^*) = \mathcal{D}_{c_{k+1}|a}$ and also comprising a second pseudonym statement with pseudonym $\mathcal{D}_{c_{k+1}}$, both of which refer to **D**, and informs the certifier that it intends to delegate a statement. The intention of delegation is signed with a Fiat-Shamir signature [FS86] based on ϕ_D^* . The channel $\mathbf{D}^{\phi_D^*} \xrightarrow{t_{10}[t_9]} \phi_{A,D} \mathbf{A}$ is established for **D** instructing **A** about its intention to delegate authority over an attribute statement. For establishing this channel, it is crucial that **D** learn the pseudonym $\mathcal{A}_{c_{k+1}|d}$ that **A** acts under towards both the certifier \mathbf{C}_{k+1} and **D** such that it can include it in the authenticated message for initiating delegation sent to **A**. For ϕ_D' we assume that only statements based on ϕ_D and a pseudonym be comprised and no further credentials be involved. **D** authenticates ϕ_D' under the same pseudonym $\mathcal{D}_{c_{k+1}|a}$ towards **A** as next step of initiating the delegation and instructs **A** through an authenticated message, comprising a delegation authorization token \mathcal{Y} , over the channel that it intends to delegate authority over the attributes contained in ϕ_D' to **A**. The authorization token \mathcal{Y} particularly includes also **A**'s pseudonym $\mathcal{A}_{c_{k+1}|d}$ and the content of the token is signed with a Fiat-Shamir signature based on the authenticated statement ϕ_D' . The pseudonym $\mathcal{A}_{c_{k+1}|d}$ needs to be authenticated by

AuthDele-Cred

$$\begin{aligned}
& \left(\mathbf{A} \xrightarrow{\phi_i, \bullet^{t_{i2}[t_{i1}]}} \phi_{C_i, A} \mathbf{C}_i, \mathbf{C}_i \xrightarrow{\phi_{C_i, B}, \bullet^{t_{i4}[t_{i3}]}} \phi_{B, C_i} \mathbf{B}, \right. \\
& t_{13} > t_{i2}, t_{13} > t_{i4}, \mathbf{B} \text{ trusts } \mathbf{C}_i, \\
& \left. \lambda(\phi_i) = \mathcal{A}_{c_i}, \lambda(\phi_{C_i, A}) = \mathcal{C}_{i, a}, \lambda(\phi_{C_i, B}) = \mathcal{C}_{i, b}, \lambda(\phi_{B, C_i}) = \mathcal{B}_{c_i}, \right)_{\forall 1 \leq i \leq k} \\
& \mathbf{D} \xrightarrow{\phi_D, \bullet^{t_6[t_5]}} \phi_{C_{k+1}, D} \mathbf{C}_{k+1}, \mathbf{D} \xrightarrow{\phi_D^*, \bullet^{t_6^*[t_5^*]}} \phi_{C_{k+1}, D}^* \mathbf{C}_{k+1}, \\
& \mathbf{C}_{k+1} \xrightarrow{\phi_{C_{k+1}, B}, \bullet^{t_8[t_7]}} \phi_{B, C_{k+1}} \mathbf{B}, \\
& \lambda(\phi_D) = \mathcal{D}_{c_{k+1}}, \lambda(\phi_D^*) = \mathcal{D}_{c_{k+1}|a}, \lambda(\phi_{C_{k+1}, D}) = \lambda(\phi_{C_{k+1}, D}^*) = \mathcal{C}_{k+1, d}, \\
& \lambda(\phi_{C_{k+1}, B}) = \mathcal{C}_{k+1, b}, \lambda(\phi_{B, C_{k+1}}) = \mathcal{B}_{c_{k+1}}, \\
& \mathbf{B} \text{ trusts } \mathbf{C}_{k+1}, \\
& \mathbf{D} \xrightarrow{\phi_D', \bullet^{t_{10}[t_9]}} \phi_{A, D} \mathbf{A}, \mathbf{A} \xrightarrow{\phi_A', \bullet^{t_{12}[t_{11}]}} \phi_{C_{k+1}, A} \mathbf{C}_{k+1}, \tag{3.20} \\
& \lambda(\phi_D') = \mathcal{D}_{c_{k+1}|a}, \lambda(\phi_{A, D}) = \mathcal{A}_{c_{k+1}|d}, \\
& \lambda(\phi_A') = \mathcal{A}_{c_{k+1}|d}, \lambda^D(\phi_A') = \mathcal{D}_{[a]c_{k+1}}, p(\mathcal{D}_{[a]c_{k+1}}) = \mathbf{D}, \\
& \lambda(\phi_{C_{k+1}, A}) = \mathcal{C}_{k+1, A}, \mathbf{D} \text{ trusts}^D \mathbf{A}, \\
& t_5^* > t_6, t_9 > t_6^*, t_{11} > t_{10}, t_{13} > t_5, t_{13} > t_5^*, t_{13} > t_7, t_{13} > t_{11}, \\
& \mathcal{F}^P \left(\mathcal{F}^R(\mathcal{F}(\phi_D, \mathbf{C}_{i+1}, \aleph_{i+1}), \mathcal{P}_D) \wedge \theta_{D, A}^{t_9} \wedge \eta_{D, A}^{t_9} \wedge \chi_{D, A}^{t_9} \right) \vdash_{\mathcal{O}_D} \phi_D', \\
& \mathbf{A} \xrightarrow{\phi', \bullet^{t_{14}[t_{13}]}} \phi_{B, A} \mathbf{B}, \lambda(\phi') = \mathcal{A}_b, \lambda^D(\phi') = \mathcal{D}_{[a]b}, p(\mathcal{D}_{[a]b}) = \mathbf{D}, \\
& \lambda(\phi_{B, A}) = \mathcal{B}_a, \\
& \mathcal{F}^P \left(\left(\bigwedge_{i=1}^k \mathcal{F}^R(\mathcal{F}(\phi_i, \mathbf{C}_i, \aleph_i), \mathcal{P}_A) \right) \wedge \mathcal{F}^{DR}(\mathcal{F}^D(\phi_D, \phi_D', \mathbf{C}_{k+1}, \aleph_{k+1}), \mathcal{P}_A) \right. \\
& \quad \left. \wedge \theta_{A, B}^{t_{13}} \wedge \eta_{A, B}^{t_{13}} \wedge \chi_{A, B}^{t_{13}} \right) \vdash_{\mathcal{O}_A} \phi' \\
& \implies \mathbf{A} \xrightarrow{\phi', \bullet^{t_{14}[t_{13}]}} \phi_{B, A} \mathbf{B}
\end{aligned}$$

\mathbf{A} to \mathbf{C}_{k+1} as part of the statement ϕ'_A . The authentication of this pseudonym and it being comprised in \mathcal{Y} allows \mathbf{C}_{k+1} for cryptographically binding the delegation credential to the correct pseudonym and thus the correct party \mathbf{A} .

Note that the model captures the annotations, pseudonyms, and security symbols in place for a delegation to be possible, however, does not make the action of instructing the delegation to \mathbf{C}_{k+1} explicit—this is done through messages sent over the accordingly authenticated channels as explained above. Note furthermore that, through the use of Fiat-Shamir signatures related to the initiation of the delegation, we can obtain non-repudiation of the intention to delegate. With pseudonyms $\mathcal{D}_{c_{k+1}|a}$ and $\mathcal{D}_{c_{k+1}}$ associated with the same party, the link to the attribute information to be delegated and held by \mathbf{C}_{k+1} is established unambiguously.

The above is sufficient for initiating a delegation and it is left as a potential future extension to the rule to allow for a further statement to be made by \mathbf{D} to further authenticate towards \mathbf{A} using the principles of Rule Auth-Cred. The formula ϕ'_D captures the parts of the credential issued originally to \mathbf{D} based on ϕ_D to be delegated to \mathbf{A} and is defined analogous to ϕ' of Rule Auth-Cred, only based on a single ϕ_i , and without requiring any trust assumptions of \mathbf{A} in the certifier \mathbf{C}_{k+1} .¹¹ For the differences to an application of Auth-Cred, this part of Rule Auth-Cred needs to be expressed explicitly again in Rule AuthDele-Cred and cannot leverage the already-defined Rule Auth-Cred.

\mathbf{A} authenticates a statement ϕ'_A towards \mathbf{C}_{k+1} , as required by the latter's authorization policy for issuing a delegation credential, which is done through an application of Rule Auth-Cred—possibly with different certifiers than those referred to in the current rule—and provides the token \mathcal{Y} with the contained Fiat-Shamir signature it has received from \mathbf{D} as proof of authorization to receive a delegation credential. Then, \mathbf{A} receives, on the channel $\mathbf{A} \xrightarrow{\phi'_A} \xleftarrow{t_{12}[t_{11}]} \phi_{c_{k+1},A} \mathbf{C}_{k+1}$, a delegation credential from \mathbf{C}_{k+1} corresponding to an attribute statement obtained through a transformation of both ϕ_D and ϕ'_D using $\mathcal{F}^D(\phi_D, \phi'_D, \mathbf{C}_{k+1}, \mathfrak{N}_{k+1})$. The channel over which \mathbf{A} receives the credential is assumed to be available, its prerequisites, which are not made explicit in the rule, must be fulfilled as per the channel transformation rules used to derive this channel. The issued credential is bound to party \mathbf{A} as its holder through its pseudonym $\mathcal{A}_{c_{k+1}|d} = \lambda(\phi'_A)$ and \mathbf{D} as its subject through the pseudonym $\mathcal{D}_{[a]c_{k+1}} = \lambda^D(\phi'_A)$ of the authenticated formula ϕ'_A .

Release of authenticated statements. For modeling the release of delegated attributes by \mathbf{A} , we require additionally a function \mathcal{F}^D that captures the issuing of a delegation credential and the inherent transformations of the authenticated statement of its recipient with the issuer, analogous to \mathcal{F} , however, taking specific aspects of delegation into account. The first argument of \mathcal{F}^D is the formula specifying the original authenticated formula ϕ_D of the delegater to the certifier on which the original credential is based, the second argument the constrained formula ϕ'_D

¹¹ \mathbf{A} is not relying on authenticated attributes vouched for by \mathbf{C}_{k+1} , rather, it gets a credential issued from this certifier. The *trusts*-relation needs to hold between the data recipient \mathbf{B} and \mathbf{C}_{k+1} because \mathbf{B} relies on attributes the certifier vouches for.

specifying which fragments of this original credential to delegate. The third and fourth arguments are analogous to the rightmost two arguments in the signature of the function \mathcal{F} . \mathcal{F}^{DR} is, analogous to \mathcal{F}^{R} in the non-delegation case, a function reflecting a term renaming of terms used in the formula to reflect the terms used by \mathbf{A} . The symbol \mathcal{P} refers, as above, to the data repository of the party indicated as its index.

The channel $\mathbf{A}^{\phi'} \xrightarrow{t_{14}[t_{13}]} \phi_{B,A} \mathbf{B}$ from \mathbf{A} to \mathbf{B} , where \mathbf{A} is annotated with the unauthenticated formula ϕ' , models the statement \mathbf{A} makes to \mathbf{B} . The definition of ϕ' is an extension of that of formula ϕ in Rule Auth-Cred in that in addition the formula $\mathcal{F}^{\text{DR}}(\mathcal{F}^{\text{D}}(\phi_D, \phi'_D, \mathbf{C}_{k+1}, \mathbf{N}_{k+1}), \mathcal{P}_{\mathbf{A}})$ representing the delegated attributes and further pseudonym statements captured in $\theta_{\mathbf{A},\mathbf{B}}^{t_{13}}$, namely those made by \mathbf{A} having \mathbf{D} as subject, can influence the derived formula ϕ' . $\eta_{\mathbf{A},\mathbf{B}}^{t_{13}}$ and $\chi_{\mathbf{A},\mathbf{B}}^{t_{13}}$ are defined analogously as in rule Auth-Cred specifying conditionally released identities and opaque identities, respectively. Note that \mathcal{P} is the repository of the party as indicated in the index.

It is a crucial feature that \mathbf{A} can make, through formula ϕ' , a pseudonym statement about \mathbf{D} , that is, having \mathbf{D} as subject. We use the function λ^{D} for referring to the pseudonym $\mathcal{D}_{[a]b}$ of the delegator who may be the subject of parts of the assertions made through ϕ' : $\lambda^{\text{D}}(\phi') = \mathcal{D}_{[a]b}$. Through the authority over attributes delegated to \mathbf{A} , party \mathbf{A} can make attribute statements about \mathbf{D} as part of ϕ' consistent with $\mathcal{F}^{\text{DR}}(\mathcal{F}^{\text{D}}(\phi_D, \phi'_D, \mathbf{C}_{k+1}, \mathbf{N}_{k+1}), \mathcal{P}_{\mathbf{A}})$.

Multiple applications of Rule AuthDele-Cred for a delegatee \mathbf{A} and the same delegator \mathbf{D} using the same pseudonyms for those and the same issued delegation credential allow \mathbf{A} to make further authenticated statements about itself and \mathbf{D} , and create corresponding evidence that they are about the legitimate pseudonym holders. It is also possible that new pseudonyms are used for the parties in further statements \mathbf{A} makes about itself and \mathbf{D} . Thus, pseudonyms, also those referring to a delegator, can be used much like in a regular credential system.

The relationship of \mathbf{D} with \mathbf{A} , expressed through the predicate trusts^{D} , expresses a different flavor of trust, relevant in the delegation context, than our standard notion thereof. It captures trust of \mathbf{D} that \mathbf{A} does not misuse the delegated attribute authority, i.e., only uses it as agreed, where such an agreement is out of the scope of the model.

Note that the ontologies \mathcal{O} being referred to in the rule are the respective ontologies used by the party as indicated in the index.

Discussion. The rule transfers authentications of \mathbf{A} with certifiers \mathbf{C}_i for $i \in \{1..k\}$ and of \mathbf{D} with \mathbf{C}_{k+1} , with an authorized delegation of parts of the latter to \mathbf{A} , to \mathbf{B} . The rule is atomic in the sense that it cannot be split into multiple rules and still retain its full expressive power of combining the transfer of multiple authentications of \mathbf{A} with certifiers while also comprising the delegation semantics.

In terms of cryptographic protocols, the establishment of pseudonyms, establishment of authentications used on prerequisite channels, and the issuing of credentials and delegation credentials are separate cryptographic protocols, and so is the proof

of a formula ϕ' using credential protocols, much of this being analogous to Rule **Auth-Cred**.

Through application of Rule **Auth-Cred**, **A** can remain non-identified and still accountable towards \mathbf{C}_{k+1} by conditionally releasing uniquely identifying data when authenticating ϕ'_A on the channel of **A** with \mathbf{C}_{k+1} . In addition, as an extension of the strong accountability features of privacy-enhanced transactions of Rule **Auth-Cred**, Rule **AuthDele-Cred** models the capability that both delegater **D** and delegatee **A** can remain anonymous yet be accountable—through the conditional release of (identifying) information about either or both those parties—towards **B** on the target channel, thereby realizing an accountability-enabling authentication transaction as discussed in Chapter 2. That is, a transaction with both strong privacy and accountability properties can be executed, while not compromising either property. This is a crucial property of the system discussed in this thesis.

The delegation Rule **AuthDele-Cred** is the only rule which allows a bullet-decorated endpoint annotation statement to move from one party (**A**) to another party (**D**), together with its bullet symbol. This does not conflict with the general principle of the Maurer and Schmid model and our model that security annotations cannot—speaking intuitively—change sides on channels. The crucial aspect here is that the statement is adapted in terms of who states it, namely **A** instead of **D**, while its subject remains the same, namely the delegater **D**, and so do the attribute assertions about **D**. That is, a different party receives the possibility to authenticate such adapted statement.

3.6 Authentication vs. Authorization

The presented channel model expresses security properties between parties through the conceptualization of channels between the parties. It captures the authentication properties of parties, but does not model authorization. However, the operation of a practical system is governed through authorization policies based on which authentications are performed. We proceed with presenting the definition of authorization used for our system.

Definition 3.5 (Authorization) *Authorization is the process of computing the rights of a party to access a resource at another party based on the statements that the first party has authenticated.*

Whenever a party **A** intends to access a resource of another party **P**, it is required to get authorized for this. The authorization policies of party **P** specify the set of possible statements of which **A** needs to proof one statement correct for getting authorized for accessing the resource at **P**. That is, there are two separate layers in the system architecture for *authentication* and *authorization*. Loosely speaking, the authentication layer is responsible for transferring authentications with certifiers to verifiers as captured in our channel model. The authorization layer is responsible for computing a decision on whether an authenticated statement, or multiple such statements, give the party access to a resource. Determining which (of possibly many

eligible) statements are to be authenticated in an interaction is a related process at the party that wants to be authorized. See Sommer [Som11] for further details on a system architecture following this design of separating the authentication and authorization layer, or the PRIME architecture [Som04b, Som05, CCS06, Som08] for an earlier and related approach. Separating authentication and authorization is a fundamental design principle for a privacy-enhanced identity management system.

In the context of the secure channel model, authorization governs the authentications a party **A** must have with another party **P** in order that the other party will grant the party access to a requested resource. The resources of a party **P** requiring authorization are thereby any of the following: a service offered by **P** or data about itself or other parties held by it, the resource representing the service of **P** of allowing for transferring an authentication with it to another party through issuing of a credential, or the resource representing the transfer of authentication in the context of delegation.

That is, parties have policies in place that govern when those parties act accordingly such that cryptographic transformations for realizing a channel transformation rule can be executed. This explains the relevance of authorization for the channel model, namely in determining which formulae need to be authenticated to make rules applicable, that is, their prerequisites fulfilled.

Because all parties have authorization policies in place to protect their data which are also considered to be resources, each release of data needs to be authorized, that is, may require the potential data recipient in an interaction to also authenticate statements to fulfill authorization policies. This leads, in most practical settings, to the use of *negotiation protocols* for the mutual release of certified data or—phrased in terms of the channel model—transfer of authentications with certifiers. We have developed a simple privacy-enhanced negotiation protocol for use in our architecture [Som11] considering also the agreement of data handling policies.

Concretely, for Rule **Auth-Cred**, each certifier \mathbf{C}_i for $1 \leq i \leq k$ has authorization policies in place that govern when a credential can be issued and thereby determine the required formula ϕ_i to be authenticated by a party **A** that wants an authentication to be transferrable to other parties.¹²

Similarly, for Rule **AuthDele-Cred**, in addition to the certifiers as above, also \mathbf{C}_{k+1} has authorization policies in place that govern the issuing of a delegation credential to the delegatee. Such an authorization policy demands a delegation request by the delegater that uniquely identifies the delegater within the scope of \mathbf{C}_{k+1} and an authentication of the delegatee following business and legal requirements, e.g., in terms of accountability of a pseudonymous delegatee. Parties **A** and **D**—users in the envisioned practical scenarios—decide, based on business or other needs, what actions to take, that is, whether and when to perform a delegation or act as delegatee. This is not modeled and governed through authorization policies of the parties.

For our system we use ontologies to capture, in a formal model, a subset of the relevant world. Ontologies are used by parties for the computation of authorization

¹²As the formula ϕ_i can be combined from multiple formulae as explained earlier, only a formula comprising partial information thereof may need to be authenticated by **A**.

decisions and the decision which credentials they can use to authenticate a statement that is to fulfill a policy. Details on the use of ontologies are presented in Chapter 4 and particularly Sec. 4.10.

3.7 Evidence and its Verification

The rules *Auth-Cred* and *AuthDele-Cred* build on the concept of certifiers authentications with which are transferred to target channels. Authentications of a party with certifiers reflect what the certifiers vouch for towards other parties that trust them for this. The rules are closely related to the definition of statement authentication and statement-authenticated channels. A key concept hereto is *evidence* for the truth of a statement. A trivial case of evidence is when a certifier trusted by the verifier directly expresses such statement towards the verifier as then no further evidence for the truth of a statement is required—the declaration by the trusted party suffices. This can be achieved with standard—and rather straightforward—identity management protocols with online certifiers that are not the focus of this thesis.¹³ More interesting to us is *cryptographic evidence* for the correctness of a statement, concretely *cryptographic proofs* based on private certificate protocols, which are the subject of Chapter 6 of this thesis.

During statement authentication, an authenticating party **A** provides evidence Ξ to a verifier **B** that a formula ϕ holds. The verifier needs to verify the statement ϕ w.r.t. the provided evidence Ξ with a well-defined procedure to assess whether the statement is true following its semantics. Upon successful completion of such procedure, **B** can be assured of the truth of ϕ . The process of generation of evidence by **A** and verification thereof by **B** is determined by ϕ and is discussed later in this thesis. Intuitively, truth of a statement ϕ means that its assertions are consistent with what the referred-to certifiers vouch for.

For the approach chosen in this thesis, evidence is created through cryptographic protocols. The cryptographic protocols we will further elaborate on at later points in this thesis are the following: establishing a pseudonym with another party, issuing a credential on a pseudonym, proving holdership of k credentials and their relations to conditionally released attribute groups, groups of opaque attributes, as well as one or more pseudonyms. The protocol for proving allows one to compute a Fiat-Shamir signature on a message using the statement that is proven. Traditional cryptographic protocols required for some channel transformations are not dealt with in this work and may have been referred to in the discussion of the rules above or discussed by Maurer and Schmid for their model [MS94, MS96].

¹³Note that the approach of an online certifier making a statement to the data recipient is not captured in those rules due to the channels with the certifier not being available as prerequisites—the rules are designed for private certificate technology.

3.8 Authentication with Credential Protocols

The semantics for the language in which formulae being authenticated are expressed is defined in Sec. 4.11. Roughly speaking, authentication of a formula ϕ is equivalent to the verifier being able to verify the evidence, that is, cryptographic proof Ξ , supplementing the formula. Recall that the formula makes statements about possibly multiple parties and about the certifiers that endorse those statements. A proof Ξ of a formula particularly is *evidence* verifiable by the verifier that the certifiers indeed vouch for the assertions as specified in the statement ϕ .

For the rules **Auth-Cred** and **AuthDele-Cred** building on credential protocols as cryptographic mechanisms, a proof Ξ is a cryptographic protocol based on a zero-knowledge proof of knowledge for the truth of the formula ϕ . We explain in Chapter 6 how such proof is generated and verified using run-time-generated cryptographic protocols as a contribution of this thesis.

One specificity is worth exploring further, namely the *trust relationships* of verifiers in certifiers, expressed through the **trusts** predicates in the model. In an oversimplified system, those can be accounted for by a verifier considering all certifiers to which it has such trust relation as certifiers statements vouched by which it accepts as valid, independent of the attributes concerned or their use.

However, trust is, in an open identity system, not a binary factor, but dependent on the context of the transaction. That is, whether a verifier trusts a certifier for making a statement about a certain attribute of some party, depends, for example, on which attribute this is and what the verifier uses the attribute for. Thus, expressing the trust relationships as a binary predicate as in our model presented so far is not sufficient for an open authentication system.

Therefore, we move the trust decisions of a verifier in certifiers from the authentication to the authorization stage of a transaction. For authentication it is therefore irrelevant for the verification of a proof Ξ for a party annotation statement ϕ whether the certifiers of the assertions in the statement are trusted by the verifier. What matters is only a cryptographically correct verification of the proof with the proper (authentic) key material of the stated certifiers. For our channel model this is equivalent to removing the **trusts** predicates from rules expressing the trust relations of verifiers in certifiers. The notion of authentication implied by such adopted system is equivalent to a more relaxed notion of authentication which does not consider the **trusts** predicates and thus trust relationships of the verifier, as well as party registration. We refer to this notion of authentication also as *cryptographic authentication*, or, just as authentication if the flavor of authentication we mean should become clear from the context. The verification process through a run-time-generated cryptographic protocol is summarized in Chapter 6 on the cryptographic protocols. It strongly connects the intuitive channel calculus with the formal semantics of the logic for expressing statements. Note that the fact whether a statement is authenticated by a party that is registered in the system is ignored for the property of cryptographic authentication as well and considered at the authorization layer.

In the authorization process following the authentication process, the trust relationships of a verifier with certifiers are considered in a fine-granular and flexible

way by the verifier when computing authorizations based on statements authenticated following the above relaxed notion of cryptographic authentication. Also, party registration, see Sec. 3.9, can be accounted for in the authorization process by having authorization policies capture requirements on credentials to be included in the statement to authenticate for establishing that a party is a registered party in the system.

This way of separating authentication from authorization, in particular with respect to trust relationships parties have with certifiers, is beneficial for open identity systems with a plurality of certifiers and verifiers as it allows for a degree of expressiveness that cannot be achieved when fixing trust decisions already at the level of authentication as is done in today's practice. Particularly, it allows a verifier to specify trust requirements in certifiers depending, e.g., on the attributes and their use. The reader is referred to Sec. 4.6 and Chapter 4 for details on how those trust requirements are expressed in the elements related to expressing data requirements of authorization policies. Through referring to ontology concepts, we can achieve what we refer to as *outsourcing of trust*, that is, deferring certain decisions related to trust, e.g., trust in parties such as certifiers, to third parties, as discussed in Sec. 4.10.

In Chapter 6 we summarize how a party computes—based on a formula ϕ the cryptographic proof Ξ of which can be verified successfully, that is, an authenticated formula in the model without considering trust in the channel model—whether a given data request is fulfilled through this formula. A formula, the cryptographic proof of which does not verify, is not authenticated.

Authenticity of public keys. Cryptographically verifying evidence based on credential protocols requires the use of public keys, e.g., those of certifiers (credential issuers) and conditional data recipients. The trust relationships related to the certification of the public keys of credential issuers through traditional PKI technology may be handled through standard mechanisms of processing certificate chains and having trusted certification authorities. Expressed differently, those trust relationships can be considered already for the cryptographic authentication. The reasons for this approach are the following: (1) we ensure technical compatibility with the existing PKI infrastructure—except for the formats of the keys to be certified—and thus simplify a deployment of our system, and (2) we remain compatible with the trust approach in the current infrastructure and thereby do not further complicate our policy system by also making generalized trust statements about those standard PKI certification authorities. Such general approach is the one possible in the related setting of trust management [BFL96, LGF03]. A generalization of this to also defer those trust decisions to the authorization computation is possible and would account for the idea that also trust in certification authorities is not binary, thereby making authorization policies more complex. This generic approach is only followed for pseudonymous credential issuers as explained next.

The certification of public keys of *pseudonymous certifiers*, see Sec. 4.4.10 for details about this concept, is based on Fiat-Shamir signatures using credential proof of knowledge protocols executed by parties known under attribute statements, instead

of on conventional signatures issued by certification authorities as in the standard case of identified issuers. Thus, the related trust decisions regarding the credential issuers (certifiers) for the public key certification need to be taken as part of the authorization decision like for a regular authentication using credentials. The cryptographic authentication comprises the cryptographic verification of the certification of the public key, like in the case of a traditionally certified public key. A pseudonymous certification of a public key and a party \mathbf{A}' obtaining such certified key can be modeled with Rule **Auth-Cred**: A credential public key of \mathbf{A} is signed by \mathbf{A} by computing a Fiat-Shamir signature on it w.r.t. the pseudonym and attribute statement it intends to be known under as evidence. Assertions based on attributes certified by multiple certifiers can be combined for the certification of the key. The signed key is then provided to party \mathbf{A}' who needs to verify the signature w.r.t. the pseudonym and attribute statement and thus knows the public key certifier by an authenticated pseudonym and attribute statement and not necessarily its legal identity. Subsequently, \mathbf{A}' can use the public key for verifying cryptographic evidence relying on this key. Each public key of a pseudonymous certifier to be obtained by a party \mathbf{A}' requires application of one instance of the Rule **Auth-Cred** to let \mathbf{A}' obtain the key authentically. Using standard means of key distribution through a key directory can be expressed as well with the concepts of our model.

Our authorization language presented in [Som11]—see Sec. 4.6 for the data-related aspects thereof—is envisioned to be mainly used such that the certifier of attribute assertions of a statement to be authenticated is specified through a statement revealing the certifier’s legal identity. Certification authorities certifying the public key of such certifier are captured statically and are not explicitly mentioned in a policy, much like in the standard PKI setting where one relies on exhaustively specified lists of trusted certifiers. The authorization language allows for making more than those levels of certification explicit in a statement, however, this is adding complexity to the policy authoring process and the gained flexibility is, in our view, not required in most scenarios we envision.

3.9 Registration and Derived Authentications

The authentications of party \mathbf{A} with certifiers \mathbf{C}_i for $1 \leq i \leq k$ in rules **Auth-Cred** and **AuthDele-Cred** and the authentication of \mathbf{D} with certifier \mathbf{C}_{i+1} in Rule **AuthDele-Cred** can be of either of the following forms, or a combination thereof:¹⁴ (1) established through out-of-band means or assigned by the certifier; or (2) established based on existing channels through the application of channel transformation rules. Practically, it is also possible to combine those cases to obtain a single authentication through channel transformation rules.

Case (1) models party \mathbf{A} getting registered in the authentication system, where an initial authentication of a statement based on out-of-band means is performed and optionally further attribute statements are assigned to \mathbf{A} by \mathbf{C} . This is modeled

¹⁴In the discussions in Sec. 3.5, Case (1) here has been split into separate cases, which we abstract from here.

through a channel with bullet-decorated annotation formula on the side of \mathbf{A} that is assumed to exist. We refer to this as a *registration* of a party in the system, and the authentication of the statement as *registration authentication*. The key characteristics of a registration authentication modeled with such a channel is that it is not transferred from existing channels through transformation rules, but is *original* in the meaning of the model. The statement of the initial authentication can, at its minimum, only be a pseudonym statement about \mathbf{A} to which further statements are bound.

Case (2) models authentication of \mathbf{A} through the authentication system, that is, the channel with bullet-decorated endpoint annotation on the side of \mathbf{A} is derived through the application of channel transformation rules based on existing authenticated channels, each of which being of either kind (1) or (2) or combinations of such. We refer to the authentication of the statement on such a channel as *non-registration authentication*, or *derived authentication*.

Based on either kind of channel, its authenticated annotation can be transferred to new channels using channel transformation rules. As per the rules **Auth-Cred** and **AuthDele-Cred**, the annotation formulae of multiple authenticated channels of any kind can be combined for deriving the authenticated target annotation ϕ' towards a party \mathbf{B} . This reflects the generic application of channel transformation rules, where \mathbf{B} can be any data recipient, also a certifier.

Within the Maurer–Schmid model, the concept of a registration is, similarly to our model, not made explicit, however, it is realized with channels from users to certifiers, having a security symbol on the side of the user, which are assumed to exist and can be used for deriving further channels, also with a security symbol on the side of the user.

Channels established following Case (1) are required for bootstrapping a system because security symbols cannot be “created” through applying channel transformation rules and need to exist originally. At least one such original channel is required per used in order to be able to derive new channels with authenticated annotation statements, that is, to be able to authenticate certified attribute statements to parties.

Based on a single registration, an arbitrary number of non-registration authentications with certifiers can be established, corresponding to authenticated channels with those certifiers, the authenticated statements of which can be transferred to other channels with verifiers using rules **Auth-Cred** and **AuthDele-Cred**. Typically, certain kinds of attributes, such as access rights to services, are assigned by a certifier that has acted as verifier of a statement before, as additional statements about the party. This reflects the standard model of a credential system where a user performs a single registration, as, e.g., in Lysyanskaya, Rivest et al. [LRSW00] or Camenisch and Lysyanskaya [CL01], resulting in a registration pseudonym and credential based on which further credentials can be obtained from other certifiers. Those works denote the party at which users register as certification authority, and in [CL01] one of its purposes is to realize non-transferability of the user’s private key and thus credentials. Both works assume for a system to comprise a central registration certifier that needs to be accepted by all users, which is—in our view—a

severe limitation for a practical system.

Security symbol. Intuitively, a bullet symbol on a channel from **A** to **B** means that the evidence provided for the corresponding statement can be verified successfully by **B** as well as that the statement the symbol decorates is being authenticated by a party **A** registered in the system. That is, (1) the evidence underlying the statement can be verified by **B**, and (2) according to **B**'s authorization policy, the statement ensures that **A** is a registered party. As a counterexample to (2), a pseudonym statement made without establishing that it is based on the private user key generated in a registration with a certifier does not receive a bullet symbol, even though its cryptographic evidence verifies correctly.¹⁵ For the credential protocols of this thesis, this means that for a pseudonym statement to be authenticatable, it must build on the same user private key as one of this user's pseudonyms created in a registration interaction with a registration certifier.

Combining multiple registrations. An interesting case captured by the model is when multiple channels representing registrations based on formulae $\phi_i, 1 \leq i \leq k$ are combined by **A** through rules Auth-Cred or AuthDele-Cred for authenticating a new statement ϕ' towards **B** while ensuring that **A** is the same on all the channels. Using multiple registrations of **A** means that additional consideration is required for establishing that the party referred to in the different resulting statements of the channels is the same party **A**. This can be achieved by a (potentially different) certifier **C'** making a statement that (a subset of) the pseudonyms related to the statements ϕ_i of the different registrations refer to the same party and this being considered in ϕ' . In order to achieve this, **A** needs to prove with additional means to **C'** that the pseudonyms it proves holdship of to **C'** refer to the same party. This can, e.g., be accomplished through revealing identifying attribute statements on both pseudonyms strongly bound to the pseudonyms. This and also the subsequent binding of the different pseudonyms to the same party through **C'** can be accomplished through private certificate protocols. The sameness of the pseudonyms can be vouched for by **C'** through issuing a so-called *binding certificate* or *binding credential*. For **A** being able to convince **B** about the sameness of the party behind the pseudonyms, it needs to include a statement into ϕ' on the sameness of the pseudonyms and proof thereof based on the binding credential obtained from **C'**. The intuition on security symbols and registrations given further above generalizes to this setting of multiple registrations being the basis for an authenticated statement. Such setting is interesting for practical credential systems as it allows for greater system openness by not mandating a single registration per user and thus giving users the choice of their preferred registration certifier.

Basing a statement a party **A** authenticates towards a party **B** on multiple registration authentications without additionally establishing party equality as discussed

¹⁵Note that in our open system, any party can assume the role of a certifier—whether it is recognized as such depends on the authorization policies of the verifiers of certified attribute statements.

results in an identity statement authenticated by one party without binding of the different parts of the formula to the same party, and thus is of limited use.

Bootstrapping of authentication. As has been explained in Sec. 3.5 on channel transformation rules, security annotations over formulae can only be transferred from prerequisite channels to target channels, however, annotations cannot be created through application of the channel composition algebra. That is, at least one channel where a party **A** is authenticated to another party needs to exist in a system for allowing one to establish further authenticated channels with **A** being authenticated. That is, the system needs to be bootstrapped by establishing initial authentications of the system participants.

Once a user has been registered, w.l.o.g. with a certifier **C** and thereby obtained a registration credential, another certifier **C**₂ can issue, after **A** has authenticated a statement towards **C**₂ using the registration credential, a credential comprising a subset of the attributes of the registration credential and further attributes, e.g., such specifying access rights to an on-line service, while being ensured that the credential be issued, in a cryptographically strong manner, to the party authenticated with a statement endorsed by **C** with which **A** was authenticated through out-of-band means.

Depending on the trust relationships between parties in a system, a single out-of-band authentication can be sufficient for each player in an identity system, or multiple out-of-band authentications can be required or at least increase flexibility of the system. In open systems like the one that is the subject of this thesis, the typical case will be that a party has performed multiple out-of-band authentications and that multiple certifiers offer the service of registering parties. Our open approach allows a party to choose which of the available registration certifiers to rely on and does not force a single registration certifier on them.

3.10 Future Work

The property of *endpoint invariance*, that is, sender or receiver invariance of a channel for an endpoint with a non-authenticated annotation formula is currently not captured. Endpoint invariance for a channel endpoint means that the other endpoint party of the channel is assured to communicate with the same party known under a pseudonym, although the pseudonym is not authenticated in the sense of the model. This property is denoted as *pseudonym invariance* by Mödersheim and Vigano [MV09b]. The endpoint invariance property can be used for modeling settings where a party does not authenticate any statement (yet), however, the other party it interacts with can be assured of always talking to the same party. An example is the setup of a one-way authenticated secure channel based on standard mechanisms between an unauthenticated user and a service provider authenticated through PKI technology. This is relevant also for our system because channels between users and service providers are such channels on which users authenticate statements using credential protocols once the channel has been established to obtain

a channel of interest. Endpoint invariance can be realized through cryptographic mechanisms, e.g., a cryptographic pseudonym without assuring that it be based on a registered user private key, or an unauthenticated encryption key pair.

Further open aspects related to the model are capturing the *revocation of credentials* in the model as well as defining a channel transformation rule modeling authentication transfer based on conventional certificates as a special case of Rule **Auth-Cred** and a rule modeling protocols with online certifiers as an extension of rule **Auth-Cred**. Delegation can be considered for those technologies by deriving according rules using the ideas of **AuthDele-Cred**.

A generalization of Rule **AuthDele-Cred** for modeling *delegation* is capturing l authentications of a delegater **D** with certifiers $\mathbf{C}_{k+1}, \dots, \mathbf{C}_{k+l}$ and delegation of authority over the corresponding statements and is left to future work to not further explode the already complicated rule. A complication here is that it must be ensured that a statement constructed from multiple delegated statements is about the same delegater. This may require the release of additional information to the certifiers in order to ensure this property using the cryptographic techniques available in our framework or the development of new cryptographic protocols. This extension may be relevant in practical settings when a user needs to delegate authority for allowing for authentications with more involved policies requiring statements about identities from multiple certifiers. An extension to support that a party is able to make statements about multiple delegaters in a single formula is less relevant from a use case perspective.

Further future work involves developing the secure channel model further towards a *stronger formalization*, analogous to what Maurer et al. have discussed in their work on constructive cryptography [Mau10, Mau11, MRT12] based on their original model.

Chapter 4

Data Representation, Logic, Ontologies, and Semantics

Our authentication model discussed in Chapter 3 as part of the secure channel model captures the authentication of *statements*, or *data statements*, by parties, the authenticating parties. Such statements made towards data recipients (relying parties) are typically endorsed (certified) by third parties, the certifiers. The authentications are governed by the authorization policies, concretely their parts expressing *data requests*, of the relying parties. Both the data statements parties make towards other parties and the data requests in authorization policies need to be expressed in a machine-processable language for a computer-implemented system. In the current chapter we discuss a logic-based formal representation of statements and data requests required for realizing a system based on our authentication model. The logic-based data modeling approach we present is technology agnostic and based on concepts related to digital identities, though, we focus on its use for private certificate systems as underlying cryptographic technology.

In a practical computer-implemented authentication system following our model of Chapter 3, further formal languages are required in addition to such for expressing data statements and requests. This comprises languages for locally expressing inputs and outputs to cryptographic protocols to be executed, e.g., for establishing identity or identifier relationships, or for specifying established identifier or identity

relationships of the party. We present the main aspects of further formal languages in our discussions of the cryptographic protocol integration of private certificate protocols into our system in Chapter 5 and Sec. 4.7 on specifying identifier and identity relationships. The language for representing data statements is not only used for statements made to other parties—we additionally express transaction logs and profiles about other parties to be stored locally through formulae in this language.

4.1 Introduction

The main driver for a commonly understood formal language used by the parties in an open system is *interoperability*. In an open system, a plurality of parties need to communicate with each other, requiring them to authenticate statements towards each other based on authorization policies of the respective other party in an interaction. The parties in a system thus need to understand, that is, make, through automated processing, sense of identity requests and authenticated statements being presented to each other. This is made possible by a common and mutually-understood formalism being used for representing data and knowledge related to the interactions between the parties. Such a formalism has the same meaning—or semantics—for all parties in the system and is a prerequisite for achieving identity management interoperability between different parties in a system.

Within the scope of a single party's system, formally represented information is subjected to multiple forms of machine processing. This comprises, e.g., storing or retrieving (finding) formulae, computing formulae based on the party's portfolio for fulfilling a data request, or associating and enforcing data handling policies to data items. Those processing instances done within a party also require a well-defined common understanding of the meaning of the data.

We use *ontologies* issued by third parties to formally conceptualize certain structural aspects or facts about our domain of interest or a given system. Such ontologies are an integral input for the automated processing of data, e.g., for computing formulae fulfilling a data request based on the party's portfolio. Ontologies are a crucial means for facilitating openness of the system and are also based on our data modeling formalism.

Using a single, unified, approach for representing data throughout the system avoids the need for mappings between different data representations at different places—each such mapping requiring a formal specification and implementation thereof—thereby greatly simplifying the overall architecture. For this reason we build upon a unified approach of data representation for representing data throughout the system. Henceforth, we refer to this as our *data model* or *data representation*, discussed in Sec. 4.4.

In terms of expressiveness, the data statement and request languages are able to model a wide range of statements about entities and at the same time to allow for the parties' privacy to be protected. The design has been strongly governed by the concept of *data minimization*, that is, the concept of the minimum possible

amount of data required by the data recipient for the purpose the transaction is executed for being revealed in a transaction. This includes the support of expressing disjunctions in formulae, predicates for reducing the amount of information being revealed about attributes instead of always revealing the attribute values, achieving conditional accountability in the setting of pseudonymous or anonymous parties, or expressing pseudonymous certifiers. Thus, the data model allows for shifting from today's prevailing paradigm of identifying parties uniquely to an attribute-based specification of the parties for stronger data minimization. Overall, the data model has been designed towards enabling user-centric privacy-enhancing identity management with unprecedented expressive power, when compared to existing approaches in research or deployed mechanisms. Delegation and openness have been considered throughout the design and use of the data model.

Simplified and rich *specification of data requests* is facilitated by supporting ontology-based abstractions, generalizing the idea of attribute-based specification of parties to certifiers of attributes in policies, and the type system. This allows for capturing a potentially large set of fulfilling formulae through a simple policy expression, thus making authorization policies simpler and more expressive in terms of the number of different statements fulfilling them, which is important for an open system. Another crucial facet of the language is the support of delegation, particularly in the setting of either delegating party or delegatee or both not being identified.

While the overall data model and the derived languages for making data statements and for requesting data are technology independent, we present technology-dependent aspects thereof related to credential system technology as this is the most advanced technology for preserving privacy and thus is the technology employed in the context of this thesis.

Chapter outline. Following the current introduction section, we present important related work in Sec. 4.2. After a brief presentation of preliminaries in Sec. 4.3, we present our generic approach to data modeling based on which concrete formal languages used in our system are derived in Sec. 4.4. In Sec. 4.5, we present the data statement language used for specifying statements to be released with credential protocols as underlying technology or further statement-type formulae required in our system, e.g., for specifying credentials. This is followed by a discussion of the corresponding data request language in Sec. 4.6. We discuss architectural aspects for an implementation of our system, presenting details on which data parties represent locally and how the data model is used for this, in Sec. 4.7.

In Sec. 4.8, we present how pseudonyms and credentials, respectively the technology-independent concepts corresponding to those, formally relate to each other and to the registration of parties, also considering delegation. Furthermore, we formally specify a renaming operation that a party needs to apply when releasing (certified) data to another party in order to ensure unlinkability and enforce properties required for the semantics.

As our data model builds on formal logic, we discuss the *calculus*, or *derivative system*, we build upon through an extension of first-order logic. We also discuss

realizing computer-implemented processing in this calculus for computing syntactic derivations therein. We discuss those aspects, as well as the relations of our work to standard calculi, tool implementations, and how to obtain a computer-implemented apparatus realizing the calculus in Sec. 4.9.

For our identity management system, we also formally model certain aspects of the structure and concrete entities in our domain of interest. Such knowledge is expressed in so-called *ontologies* that parties use for processing formulae in the calculus. We explain the types of ontologies of our architecture and how they are used in computer-implemented processing of formulae. Particularly, we elaborate on the concept of *trust outsourcing*, that is, giving third parties that are trusted for this purpose authority over certain definitional aspects of the structure and concrete facts of the system and its parties. Ontologies and trust outsourcing are discussed in Sec. 4.10.

Finally, we present the formal *semantics* for statement-type formulae based on the data model in Sec. 4.11 which globally defines the denotations for such formulae in the system.

4.2 Related Work

As this chapter takes a comprehensive approach to formally modeling identity data and related trust aspects for an open identity management system, related work from different domains needs to be addressed. We next relate our results to some of the existing relevant work. Further details on the relation thereto is presented later in the chapter in the appropriate context.

The work presented in this chapter is a substantial extension of earlier results of the author [Som11, BCS05, CSZ06a, HS06], particularly in terms of expressiveness and ontology support. Work on this has started as an effort related to the architecture for a privacy-enhancing identity management system in the PRIME project [Som04b, Som05, CCS06, Som08]. This line of work has, in terms of access control languages, built on Bonatti and Samarati's work [BS02]. The author has contributed to extensions thereof towards data-minimizing access control based on credential systems as underlying protocols [ACK⁺10].

The author has introduced the basis for attribute-based access control while, at the same time, obtaining the property of requester accountability in earlier work [BCS05], as a basis for anonymous yet accountable access control. The language presented therein already supports the grouping of attributes into conceptual units which has been presented in a more explicit way in the follow-up work on a privacy-enhancing data request language [CMN⁺10]. As specific technology for realizing the combination of the anonymity and accountability properties, these prior works also build on private certificate protocols. Further work by the author defines a framework for privacy-preserving identity management [CSZ06b, CSZ06a] based on various technologies and thereby builds on the language of Backes et al. [BCS05]. This work can be seen as introducing some of the language concepts of this thesis as well as the protocol interfaces of Chapter 5. The earliest attempt for expressing a

data statement to be released in the context of an early variant of the *Identity Mixer* anonymous credential system [Cam03] was realized directly through the application programming interface (API) of the protocol implementation which was feasible due to the limited identity semantics the credential system implementation supported.

The author and Hogben were, to the best of our knowledge, the first to use ontology-based reasoning for privacy-enhancing identity management, particularly for outsourcing certain aspects of specification to third parties, in 2006 within the PRIME project [PRI08, HS06]. They have implemented ontology-based reasoning in a prototype using W3C's RDF [ME04] and OWL [MvHE04] languages for expressing ontologies, policies, and data statements. This work by Hogben and the author [HS06] can be seen as an early predecessor of the ontology-based work presented in this thesis.

Other approaches for authentication and distributed access control in open systems build upon formal logic for specifying attribute requirements for distributed access control, while using a technology-agnostic language [AF99, LGF00, LGF03, GBB⁺06, BBG⁺07]. Notably, some works employ logic with non-standard properties, e.g., linear logic to express consumption semantics of credentials [GBB⁺06, BBG⁺07].

An important related field is *trust management*, which discusses authentication in open distributed systems. The trust management concept has been put forth in the first publication on the topic by Blaze et al. [BFL96]. Notable early trust management systems are *PolicyMaker* [BFL96, BFK99, BFS98] and the *KeyNote* trust management system [BFIK99]. Work on trust management has introduced the concept of delegation of attribute authority in the sense of delegation of trust in allowing parties to formally define in an expressive way which parties they trust for making statements about attributes of users [LGF00, LGF03]. This is also one of the main themes of our work as it is crucial for an open system. Though, parties in those trust management systems have a unique identity, e.g., public key, to which attributes are associated. Thus, privacy is not at all considered in this line of work, which results in a simpler formalism compared to our approach.

Further noteworthy approaches to trust management and distributed authentication include the logic-based distributed authorization language *SecPAL* [BGF06, BFG10], which not only supports delegation of trust, but also delegation of authority in the common meaning. A later language related to SecPAL is *DKAL* [GN08a, GN08b].

Our work realizes the most crucial functionality of trust management systems, namely trust delegation, though, in a different way than trust management systems do. Furthermore, our system is capable of protecting user privacy through data-minimizing interactions, while retaining user accountability.

Another related field is that of *trust negotiation*, the process of two parties mutually exchanging certified attribute information as governed by policies protecting this information until a policy for an initial request triggering a negotiation protocol is fulfilled. This concept has been introduced by Winsborough et al. [WSJ00], further discussed by Winsborough and Li [WL02, WL04]. Those results on trust negotiation do, like the trust management work, not consider privacy. Later work by

Li, Li, and Winsborough [LLW05, LLW09] has extended trust negotiation such that the protocols consider data minimization. Trust negotiation systems do not have the powerful mechanisms for trust delegation as trust management systems, which can be seen as a major inhibitor in practical open systems with a large number of certifiers.

The data request and statement languages we define can be used as the formal language foundation for the data exchange part of a privacy-enhancing trust negotiation system. A trust negotiation system may require that multiple received formulae be considered in a single evaluation of policies protecting a resource, for which purpose our globally valid semantics is beneficial. Non-trivial problems needed to be overcome for our privacy-enhanced system to achieve this. Also, we have defined a simple algorithm for trust negotiation based on our approach to data representation [Som11]. Through leveraging the properties of our formal data model and semantics, this algorithm can achieve stronger data minimization than previous approaches and combine privacy and accountability.

Additional relevant work can be found in the areas related to logic and calculi as well as ontologies and is reported in the respective sections of this chapter.

4.3 Architectural Preliminaries

We next discuss architecture aspects for an implementation of a user-centric authentication system realizing the secure channel model presented in Chapter 3. The basic idea is that data in a system is represented through logic-based *formulae* held and communicated by parties. Furthermore, formulae can have underlying (*cryptographic*) *material* associated, e.g., tokens that allow the party to prove (parts of) such formulae to other parties, or a ciphertext or cryptographic commitment corresponding to elements referred to in a formula received from another party, or a proof of a formula.

A party has a *repository* or *data repository* comprising all data the party holds in its formalized representation. The repository consists of the identifier relationships and identity relationships of the party, profile data the party holds, and the data track of the party.

The *portfolio*¹⁶ of a party comprises the subset of the repository which is required by the party for making data statements, proven correct through corresponding (cryptographic) proofs, to parties, or—expressed in terms of the channel model of Chapter 3—transferring authentications to parties. It comprises the identifier relationships, identity relationships, and object specification formulae for opaque and conditionally released identities, both of which are part of the data track that the party has created in interactions with other parties, with their associated metadata.

Those architecture concepts apply to any party in a system, regardless of whether it is a user, service provider, certifier, or other party. From this perspective, the underlying architecture is symmetric by not distinguishing between different parties,

¹⁶The term “portfolio” has been adopted from prior work on user-centric privacy-enhancing access control by Bonatti and Samarati [BS02].

respectively. It also facilitates the approach that parties act under certain roles, e.g., data provider (user), certifier, or relying party, instead of statically being of such type. In Sec. 4.7, we provide details on the repository and portfolio of a party based on the approach to data representation introduced in this section.

4.4 Data Model and Concepts

In this section, we present the *data model* underlying the formal representation of all kinds of data and knowledge as well as data requests required for our system. The section proceeds by introducing crucial *concepts* and our approach for formally representing them in the data model.

The model has sufficient expressiveness to satisfy, from a data model perspective, many use cases we have in mind for user-centric privacy-enhancing identity management. The fragment of first-order logic with extensions will be introduced in a step-by-step manner, with explanation of the underlying concepts and examples for illustration.

Our contribution is a language that allows one to express identity information about entities in a general way and that is particularly suitable for use with private certificate systems, today's most privacy-protecting mechanism for authenticating users to other parties, as data exchange technology. As already mentioned, we stress again that a concrete data model as we propose is a necessary precondition for a deployment of such private certificate systems in practice because an expressive and machine-processable representation of the identity data with clear semantics is required for integration with authorization and trust negotiation frameworks.

4.4.1 Basics

Our approach of data representation builds on typed first-order logic and is based on *formulae* in such a logic. Following concepts of first-order logic, things of interest are referred to using notational elements such as constant or variable terms, function symbols, expressions in the form of recursively composed terms, or predicates as is standard.

For the foundational mechanism for data representation, we assume that data about parties are captured as attributes of *identities*, *conditionally released identities*, *opaque identities* and *identifier objects* referred to through constant or variable terms. Henceforth, all those are referred to as *objects*, unless a distinction is required. Each object *o* comprises *attributes* and an attribute *a* of object *o* is referred to as *o.a* through qualification of the object with the attribute using the “.”-notation. The “.”-notation used for addressing attributes of an object is a shorthand notation for a function that realizes the object in first-order logic as explained in Sec. 4.11. Both the object and attribute are terms of the language, either constant terms or variable terms. The attribute reference *o.a* refers to the value of the attribute with name *a* of object *o*.

Later in this section we introduce a set of *reserved*, or *special*, *attributes* of objects that have predefined meaning required for the purpose of facilitating identity management, while any other attributes but those, are definable and denoted as *user attributes*.

As in standard first-order logic, a *constant* refers to a concrete object in an interpretation of a formula, while a *variable* is to be thought of as a variable in standard first-order logic, where both free and bound variables are permitted.

As we use a sorted logic, terms are *typed* through the logic and thus have assigned “built-in” types. We use the $::$ -operator as is standard for indicating the type of an element. Predicates and functions have a signature specified through the types of their arguments, and expressions are composed of terms through which their type is derived. The signature of predicates and functions is specified using the standard notation $R_{\beta_1 \times \dots \times \beta_k}$, with R a predicate or function symbol and $\beta_i, 1 \leq i \leq k$ being the types of the arguments.

Our formulae are built from *predicates* for expressing relations between attributes of objects, constants, variables, or—more generally—expressions. We support a set of *relation predicates*, depending on the data types of the arguments. This comprises **Eq**, **Neq**, **Lt**, **Leq**, **Geq**, and **Gt** which are the standard relations on totally ordered sets with their standard meaning. Negations of each of those can be expressed—using their standard semantics—with a corresponding predicate: The negation of ‘less than’ can be expressed through ‘greater than or equal,’ for example. Details on the available predicates for the supported argument types are presented in Sec. 4.4.11 on the type system.

Note that we may, for brevity, use the same predicate term for expressing relations on arguments of different data types by *overloading* the predicates for the different argument types in our notation. For example, the predicate **Eq** is used for expressing equality for arguments of any of the data types with the common equality semantics. Formally, the predicate needs to be referred to as, for example, $\text{Eq}_{\text{int} \times \text{int}}$ in case of the predicate over the integers. Thus, predicate overloading is only a simplification of notation, instead of always making the argument types explicit.

Regarding notation, we note that in this thesis we use fonts and styles for differentiating different kinds of elements of the data representation language, which should become clear from their use. On another dimension, we may, with such notation refer to the language itself or a meta-language for discussing the language and related processing over formulae expressed in it. For example, in the meta language, we may use meta-variables that may correspond to a concrete variable or constant in the concrete language. The difference between language and meta-language should be clear from the context.

Example 4.1 (Predicate) *The following is an example for referring to the attribute named `firstName` of identity `c1234` and stating that it is equal to the individual constant “Jane”. The notation in the second line shows the simplified notation*

with omitted types from the predicate signature.

$$\begin{aligned} & \text{Eq}_{\text{str} \times \text{str}}(\text{c1234}.\text{firstName}, \text{"Jane"}) \\ & \text{Eq}(\text{c1234}.\text{firstName}, \text{"Jane"}) \end{aligned}$$

4.4.2 Connectives and Formulae

Two or more predicates are connected to a more comprehensive *formula* through the standard *connectives* \wedge and \vee of first-order logic using infix notation. We allow for using parentheses in the standard way for setting precedences that deviate from the built-in precedences of the language, which are based on the standard ones used in first-order logic, that is, conjunction binds more strongly than disjunction.¹⁷

Example 4.2 (Simple formula) *A simple formula using a disjunction over attribute predicates is given next:*

$$\text{Eq}(\text{c1234}.\text{firstName}, \text{"Jane"}) \vee \text{Eq}(\text{c1234}.\text{firstName}, \text{"John"}) . \quad (4.1)$$

Particularly the possibility of the \vee -connective greatly improves our approach for representing data in terms of data minimization functionality compared to the standard name-value pairs widely used today. Furthermore, the formula-based language is able to express parties' data together with certification metadata for the data. This is useful in terms of integrating both data and trust aspects related to the data in a single model and allowing for policy decisions based on both.

Formulae expressed in our logic are the entities used to represent data or requests thereof or other aspects of the system. A formula can be used for representing data or requests in different places, such as in an identity relationship, in a data track entry, in profile data, in a data request, in a data statement made towards a party, or a data request specification in an authorization policy. A formula thereby expresses two conceptually different kinds of data: *identity data* related to one or more subjects and related *metadata*. The identity data comprises information on attributes related to the subjects of the formula, while the metadata comprises information on the certifiers of the identity data, the temporal validity of certification, and possibly other metadata. All of this forms a unit, the formula expressed in our data model. Note that different syntaxes apply to different kinds of formulae as explained later in this work.

4.4.3 Functions of Parties

In the context of an identity and identifier object, a party can have a defined *function*, where functions are *subject* and *holder* and for an identity additionally also *certifier*. For a conditionally released identity, a party can have the function of *conditional recipient*. The subject and object terms are, for example, used to relate the subjects and objects of multiple identities or identifier objects used in a

¹⁷For a simplified language, one can use prefix notation for expressing formulae and precedences and suffice with parentheses for only delimiting the scope of a logical connective symbol.

formula with each other through equality, thereby establishing that they comprise information about the same party.

Worth noting is that the *holder* and *subject* attributes are not directly referring to cryptographic keys at the level of the data model, but relate to the identity management concept of a pseudonym, one of possibly many identifiers of a party. Cryptographic keys are only one technical means for realizing those abstract concepts through using cryptographic mechanisms, such as credential systems. Protocols with on-line certifiers do not require keys for implementing those concepts, but can do so directly based on the attribute values. Our technology-agnostic way of modeling leads to some of the complications with respect to holder and subject functions.

4.4.3.1 Subject

The *subject* of an identity or identifier object is the entity or party which the data represented through the object is about. The object is associated with a subject through a *subject term* of type *pty* assigned to the *subject* attribute of the identity. We establish the convention that the *subject* attribute, being a reserved, or special, attribute, be available for each identity and identifier object. The following example shows an equality statement being expressed over the subject of identity *c* and a constant term, that is, the release of the subject term.

$$c.subject = \text{"user4567"}$$

We note that the subject of an identity is not necessarily the party using the identity, that is, releasing attribute predicates over the identity, e.g., in delegation use cases the subject is different from the delegatee.

Also note that there are multiple different uses of the $=$ -symbol in this thesis which we do not distinguish notationally as the different uses should be clear from the respective context. The use above is as object equality symbol as part of the formal language. Other uses include the that of equality symbol of the meta language about out formal languages.

4.4.3.2 Holder

The *holder* of an identity or identifier object is the party that has special privileges with respect to the identity that no other party has, allowing the party to make statements over the object using security tokens or other information it holds to prove the statements correct. In other words, the holder can be viewed as the owner of the object that can legitimately make use of it for releasing (certified) data and prove holdership of it. Holdership is independent of the party also being the subject, although those functions often coincide, particularly for standard use cases of private certificates. Delegation is a notable case when the holding party is different to the subject. Analogous to the subject, the holder of an object is referred to through a term of type *pty* being the value of the *holder* attribute of the object.

From a technical perspective, the holder of an identity is the party that holds the metadata, such as certificates, private keys, or other data, allowing it to make certifier-endorsed statements about this identity or have them made by the certifier, depending on the protocol. Analogously, for an identifier object, the holder is the party that holds the metadata to establish holdership of the identifier object towards another party.

An example of a holder of an identity is the party **P** that holds an identity relationship based on a private certificate that has been issued to **P**. Only this party **P** is able to make statements about the identity of this identity relationship, proving them correct using the certificate and its private key.

Another example is an identity based on a relationship with an on-line identity provider. The prominent protocols where an on-line identity provider asserts attributes about its users are another possible underlying technology for an identity. The holder is the party who can request the identity provider to make identity statements about the identity to other parties, that is, it can use the identity. The holder has private information, e.g., a password or authentication key, associated with the identity that allows it to authenticate at the identity provider and ensure that only it can make use of the identity by having the identity provider make statements about it. The security depends on the strength of the authentication mechanism to authenticate with the identity provider.

4.4.3.3 Certifier

The certifier is a term referring to the party certifying the identity, that is, vouching for its attributes. Details on this are presented below in Sec. 4.4.10.

4.4.3.4 Conditional Recipient

For a conditionally released identity used for modeling conditional release of data, a party can have the function of *conditional recipient* of the data—see Sec. 4.4.5 below for details.

4.4.4 Identities

A foundational concept in our formalism for representing identity data is the *identity*, a named function from a set of attributes to their typed attribute values.

Definition 4.1 (Identity) *An identity is a function from a set of attributes to a set of typed attribute values.*

The function representing an identity is, following the standard definition of a function, equivalent to a named set of tuples, each comprising an attribute name and a typed attribute value.

Example 4.3 (Identity) *We give an example identity c1234 comprising the attributes firstName, lastName, and income, indicating also the types.*

$$\begin{aligned} \text{c1234} = \{ & (\text{firstName} :: \text{att}, \text{"Jane"} :: \text{str}), \\ & (\text{lastName} :: \text{att}, \text{"Doe"} :: \text{str}), \\ & (\text{income} :: \text{att}, 3000 :: \text{int}) \} \end{aligned}$$

The example relates the attributes firstName and lastName to string constants and income to an integer constant.

Note that using the function paradigm for associating attributes of an identity with constants implies that equality is expressed between an attribute and its associated constant. Using relations other than equality is possible through the operator extension to our data model presented in Sec. 4.4.17.

Identities are referred to by terms of type *id* of the language, where such term can be an individual constant or a variable, e.g., the constant *c1234* or the variable *C*.

The use of the logical *connectives* allows one to build comprehensive formulae in our data model, e.g., for making (data-minimizing) statements about parties or requests for such. The example formula (4.1) on page 89 shows how to express a disjunction over two predicates expressed over the same identity.

An identity is used as a conceptual *grouping* of attributes, as it often occurs in real life. For example, government-issued credentials such as passports, driver's licenses, or residence permits each comprise a group of attributes relevant in their respective context. This grouping provides additional semantics to the contained attributes of the identity by stating that those attributes belong together. For example, both an account balance and a currency attribute for a bank statement need to be grouped together, otherwise they will not have the intended meaning of denoting the balance of a given account in its associated currency. Further below, we introduce the *type* of an identity expressed through a reserved attribute as a means of associating further meaning with an identity and its attributes.

In our credential-based protocols, an identity corresponds to a credential, or private certificate based on the SRSA-CL scheme [CL02b] or BL-CL scheme [CL04], that a party holds.

A given identity is immutable, that is, it cannot change over time within a concrete system as a prerequisite for a monotonous logic underlying the language. Changes to an identity are implemented by establishing a new identity with the changed information and rendering the to-be-changed identity obsolete through attached metadata governing its use in the system. Obsolete identities are not any more used for any processing in the system—cryptographically, an obsolete identity of an identity relationship can be revoked. Making an identity obsolete and enforcing this depends on the kind of identity and its uses. The change of objects over time is not modeled formally through the logic, because a more powerful logic required for expressing this does not provide a substantial improvement of functionality for our purposes, though, leads to complications.

A reserved attribute of an identity is a *unique object identification number* of the identity. Such number must be unique in the scope of all identities of a given type issued by a certifier. We prefer the generic concept of unique identification number to that of a serial number as it does not reveal information on the number of issued identities, though, we use serial numbers in this thesis for the same purpose. Note that neither number should be revealed in practical use cases, it is rather intended to serve the purpose of showing the (non-)equality between identities without revealing an identifier.

Relation to related work. The term *card* is used for a related concept in Camenisch et al. [CMN⁺10], namely the concept of an identity as defined here together with (cryptographic) tokens endorsing it. Various other works in the literature body use the term *credential* for referring to a similar concept. Other semantically meaningful names to refer to the identity concept are *attribute group* or *attribute set*. We chose the term *identity* for its genericity as well as the fact that it resembles the concept of *partial identity* which is well established in the privacy-enhancing identity management research community [PH10], see also Sec. 2. A partial identity is commonly known as a part of a party's complete set of attributes it holds about itself that it exposes to another party in an interaction or a set of related (linked) interactions, whereby an identity in our definition captures a part of a party's attribute information, but not in the context of the identity's complete comprised set of attribute information being revealed to another party.

Multiple earlier works where we used the concept of an identity to group attributes in a privacy-enhanced identity management context using private certificates or other technologies have been published [BCS05, CSZ06a, CMN⁺10, Som11], with Backes et al. [BCS05] being the first in this series of works introducing the idea. Camenisch et al. [CMN⁺10] and the author [Som11] have already proposed the technology independence of the concept while [BCS05] was focused on private certificate protocols for achieving the strong combination of accountability and privacy.

Use of identities. Identities are used for a plurality of purposes, of which we present some next. An identity may characterize a party in terms of the party's attributes of its *civil identity* (or *legal identity*), such as its name, address, date and place of birth etc., other *assigned attributes*, such as the name and grade of an online course a user has completed, or *assign rights* to the party, e.g., specify the rights of the party for accessing an online resource such as for a subscription to an online newspaper or movie rental store. Although all those cases are different in terms of what parts of the identity of a party are concerned, there is, from a technical perspective, no need to handle those cases of the identity of a party (in terms of attributes) in the strict sense and rights assigned to the party differently. Thus, we subsume all of those into the concept of identity and its attributes. This gives rise to a wide meaning of the term *identity* in the data model. In this thesis, the use of the term identity should mostly become clear from the context of use, otherwise, we may explicitly clarify it.

Indirection of attribute association. From an identity management perspective, the concept of associating attribute information with identities and identities with entities—in particular people—has the major advantages over associating attributes directly with parties that parties can be assigned multiple (different) attribute values for the same attribute and that attributes are grouped. The possibility of having different attribute values for an attribute is commonly leveraged in today’s Web interactions by users, e.g., by using pseudonyms and picking different names for different accounts resulting in the various pseudonyms people can have in different contexts on the Web, or by releasing different attribute values for other attributes, e.g., a date of birth, where no purpose of data collection for such attribute is evident to the user. This is foundational to privacy-enhanced identity management in general, see discussions in related work [PH10] for a detailed account.

Subject and certifier. An identity technically pertains to exactly one *subject*, the subject of the data represented through the identity. The subject is indicated with the *subject* attribute of the identity.¹⁸ An identity contains an attribute *subjectIdWithCertifier* comprising the *subject identifier* under which the identity’s subject is known by the certifier. This identifier is the identifier of the subject with the certifier and can be used, for example, for the revocation of the anonymity of a party carrying out a data release transaction. A formula over multiple identities may make statements over identities with different subjects, that is, talk about different parties which can, for example, be useful in scenarios with user-centric delegation where one needs to express data about the delegater and the delegatee within a single formula.

An identity is also associated with at most one *certifier*, that is, an entity (e.g., a single party or a logical party comprising multiple physical entities) who vouches for the identity and attribute information it comprises.

Another view of identities being part of identity relationships is that such an identity is a view the certifier of the identity has about the subject and that it has decided to vouch towards other parties, so-called relying parties. This represents well what an identity actually is, namely what a party who makes a statement about another party claims about this party. Thus, this view is not about the actual correctness of the attributes with respect to any real-world reference attributes, e.g., the ones in a party’s government-issued credentials such as their passport, rather it is about claims someone, who is trusted by others for this, makes about someone else. Depending on the claiming party and its policy of validating attributes it vouches for, those claims may have a certain degree of attribute assurance and thus be suitable for practical purposes of identity management for other parties, such as service providers relying on those claims for given purposes. Actual correctness of the attributes with respect to the real attribute values, such as the civil identity of a

¹⁸The concept of subject corresponds to the concept of *data subject* of the European data protection Directive 95/46/EC [Eur95] in case the subject is a user and only data about a single person is represented in the identity. This legal term does not apply in case the subject is a service provider. Note that we use the term in its technical, that is, less constrained, meaning in this chapter unless explicitly stated otherwise.

user, largely is not a technical question, but rather a question of processes executed and verification performed by the certifier when establishing the attributes of a party whose attributes it intends to vouch for as well as of this information being accessible to the other parties, that is, relying parties.

4.4.5 Conditionally Released Identities

Conditionally released identities are used in our formal language for modeling the concept of conditional release as discussed by the author earlier [Som11] and originally presented by Bangerter et al. [BCL04]. A conditionally released identity is similar to a regular identity in that it is a function that maps attributes to their typed values and thereby represents a set of tuples associating attribute names with typed attribute values. Statements can be made about the attributes of a conditionally released identity through predicates, e.g., by relating them to attributes of regular identities in a formula. The attributes *condition* of type **str** and *recipient* of type **pty** are reserved attributes of conditionally released identities. The former expresses the release condition, the latter the conditional recipient, both explained in Sec. 2.7.

As is also true for identities, the tuples related to conditionally released identities are not obtained by the data recipient during a data release interaction the conditionally released identity is created in—the data recipient only learns the predicates expressed about it. Only once, and if, the associated condition is fulfilled after a successful data release transaction, the recipient learns the tuples comprising the conditionally released identity.

We give a fragment of a formula in (4.2) to show the use of conditionally released identities within a formula to be released.

$$\begin{aligned}
 & \dots \wedge \text{tid.subject} = \mathbf{t} \wedge \\
 & \quad \text{Eq}(\text{tid.unique_name}, \text{"Swiss_revocation_services"}) \wedge \dots \\
 & \dots \wedge \text{Eq}(\mathbf{e.serialNumber}, \mathbf{c.serialNumber}) \wedge \\
 & \quad \text{Eq}(\mathbf{e.condition}, \text{"Misuse_of_service"}) \wedge \\
 & \quad \mathbf{e.recipient} = \mathbf{t} \wedge \dots
 \end{aligned} \tag{4.2}$$

The example shows how the attribute *serialNumber* of conditionally released identity **e** is specified to be equal to the attribute *serialNumber* of the identity **c** without revealing the latter. Typically, the predicate **Eq** for the data type of the attribute being referred to is used for relating attributes of conditionally released identities to attributes of certified identities. Furthermore, the example shows how the conditional release condition is expressed through the reserved attribute *condition* and the conditional recipient through the reserved attribute *recipient* of **e**. The recipient is specified through term **t** being further specified through another identity **tid**.

This approach of specifying the attributes of conditionally released identities by relating them to the attributes of identities representing private certificates has first been put forth by Bangerter et al. [BCM05] in the context of a language at the cryptographic level specifying data statements to be released—Backes et al. [BCS05]

and the current work are extensions of those earlier results using a more expressive language.

When using the technical realization of credential protocols, a conditionally released identity corresponds to a ciphertext encrypting a tuple of attributes using a verifiable encryption scheme such as that of Camenisch and Shoup [CS03] or Camenisch and Damgård [CD00].

A conditionally released identity does not have a subject or holder attribute because it may comprise attributes related to different subjects, e.g., attributes of the delegater and delegatee may be present in a single conditionally released identity, and the subject or holder should become clear from the formula the conditionally released identity is specified through, thus need not be made explicit in the conditionally released identity itself.

4.4.5.1 Discussion

The choice of again reverting to the identity concept for expressing the conditional recipient of the conditionally released identity is motivated as follows: First, it conceptually fits the idea of using identities to specify attribute statements about parties, and thus is integrated into the model and derivations over it naturally by using the same language elements. Second, it allows for specifying in a flexible way a set of parties as possible recipients through appropriate specification of the predicates in a data request, e.g., through using ontology concepts, which is particularly useful for giving choice over one of multiple data recipient parties to the data releasing party in a data request of a policy. Particularly, this allows for specifying the conditional recipient through properties instead of attributes of its identities, if an ontology modeling this is defined.

Conditionally released identities are typically used in addition to other kinds of identities in data release protocols for establishing revocability of anonymous interactions through escrowing identity information that can identify the party once being recovered. An actual revocation of anonymity—the recovery protocol—requires additional protocol flows, depending on the exact scheme being used.

Anonymity revocation. In systems where users can be pseudonymous or anonymous in an interaction, legal regulations or specific interests of parties may require that anonymity or pseudonymity be revokable under well-specified circumstances. We have defined a reserved attribute *subjectIdWithCertifier* for identities that is set during creation of an identity relationship to the subject identifier the party is known under to the certifier at the time of the creation of the identity relationship. This approach of introducing a new attribute for modeling the subject identifier under which the subject was known when the identity relationship was created avoids the need to revert to more powerful logic for expressing this meaning. For realizing revocability of a specific transaction, the *subjectIdWithCertifier* attribute of an identity referred to in the data formula to be released is conditionally released to the data recipient and can be obtained only by the specified conditional data recipient (trustee) once, and if, the associated conditional release condition gets fulfilled. The

actual revocation of the anonymity, that is, obtaining the subject identifier, can be carried out by the trustee if asked so by the data recipient or a third party and after verifying that the condition holds. With this pseudonymous identifier it is, depending on the setup, possible to obtain further, possibly identifying, attribute information of the party from the certifier of the identity.

As a specific case, the attribute *subjectIdWithCertifier* can be the identifier the party had with the registration certifier from which the registration identity for this registration domain has been obtained—see Sec. 4.8 for what registration means in the data model.

It does clearly not make sense to conditionally release a pseudonym value with the party one interacts with as it has no benefit to the interaction partner to be able to obtain this identifier conditionally. More generally, any technically feasible combination of attributes can be conditionally released by a party (as required by a data request), possibly to multiple conditional data recipients in a single transaction. Depending on the technology being used, the releasing party need not execute a protocol with the conditional recipient. Attributes of the basic data types and *idf* can be conditionally released, while *pty*-typed attributes cannot. See Sec. 4.4.11 for details on typing in our data model.

Formulae containing references to conditionally released identities are applicable for making data statements to other parties, expressing the requests hereto, and storing those formulae in the data track and profile data. Though, such formulae are not applicable for modeling data in identity relationships.

Further consideration needs to be given to specifying conditionally released identities in data requests comprising disjunctions. It is crucial that it be ensured that, once the conditional recipient obtains the attribute values of the conditionally released identity, it can be determined which identity an attribute is from. For example, it may be crucial to know whether a unique identification number of an identity is from a Swiss electronic identity card or a Swedish one, as otherwise it is not clear which authority to request further attribute values from for a de-anonymization. Such disambiguation can be done, for example, through an attribute of the conditionally released identity that indicates, as a string, the identity an attribute is from, in the case of using disjunctions in the specification formula.

4.4.6 Opaque Identities

Opaque identities are used for referring to attribute groups which a party holds and can make statements about, while not necessarily revealing the attribute values to its communication partner. That is, an opaque identity is a handle to a group of attributes, where each attribute can be related to other attributes referred to in a formula. An opaque identity is syntactically represented like the other kinds of identities through qualifying the term referring to the opaque identity through the “.”-notation with the names of its attributes. Analogously to an opaque identity, an identity corresponds to a function specifying its attributes.

An opaque identity is typically used in a way that the values of its attributes are specified through relating them with other attributes or predicates over such.

This way of specifying an opaque identity is done in a formula specifying data to be released. Once the formula has been released and proven correct with a cryptographic protocol, the opaque identity is established and its attributes fixed. For the communication partner, the opaque identity is a handle to the corresponding attribute set, while not learning the attributes unless they are explicitly revealed.

Once established, the attributes of an opaque identity can be referred to in a formula specifying a new identity to be established in order to determine their values without the certifier learning them, or they can be referred to in further released formulae, e.g., for proving equality with attributes of other identities of previously released formulae.

Example 4.4 (Opaque identity) *Let the formula fragment in (4.3) be the fragment of a formula in which the opaque identity `oid` is specified by relating its `lastName` and `firstName` attributes to the correspondingly named attributes of the electronic driver's license `edl`.*

$$\begin{aligned} \dots \wedge \text{Eq}(\text{oid}.\text{lastName}, \text{edl}.\text{lastName}) \wedge \\ \text{Eq}(\text{oid}.\text{firstName}, \text{edl}.\text{firstName}) \wedge \dots \end{aligned} \quad (4.3)$$

Once the formula has been released to its recipient and proven correct using a corresponding cryptographic protocol, `oid` can be referred to in other formulae and protocols.

4.4.7 Identifier Objects

A *pseudonym* is a name for a party following our definition of Sec. 2.3. We express pseudonyms in our language through the concept of *identifier objects*, which are entities represented in a syntactically similar way to identities. Identifier objects equally comprise attributes that can be referred to by qualifying the term `p` representing an identifier object through `.` to refer to its attributes. Any identifier object has the following reserved, or special, attributes with the indicated types:

$$\text{holder} :: \text{pty}, \text{subject} :: \text{pty}, \text{subjectId} :: \text{idf}.$$

The attribute *subjectId* is a constant term of type `idf` used as the identifier of the party, while *holder* and *subject* are of type `pty`, referring to the holder and subject, respectively, that is, the parties that may make use of the identifier object and the subject the party which the identifier object is about, much like for an identity. The value of *subjectId* represents the pseudonym being modeled.

Conceptual differences to identities are that identifier objects can only be released to, by proving holdership, the parties they are established with and that they have an exhaustively specified set of allowed attributes used to express their properties, while identities can be used with any verifying party and can have sets of user attributes associated with them as induced by their types in addition to the reserved attributes. Due to these differences, identifier objects and identities cannot be cleanly modeled with a single concept, though, their syntactic modeling is closely related.

An identifier object is not only relevant for the use case where a user communicates with a service provider and uses the identifier object in this communication to refer to itself. In a scenario where two service providers interact to exchange customer data, the concept of identifier objects equally applies. Though, the trust model is different in that in the former case, cryptographic protocols are employed to enforce the correctness of subject identifiers while in the latter case, the service providers need to trust each other in using correct identifiers. As the concepts are equal—a party talks about another party, possibly itself, towards another party—we use the same concept for modeling, while protocols with considerably different properties and underlying trust models can be used for implementing the concept of identifiers of parties. We focus on cryptographic pseudonym and credential protocols in this thesis.

Identifier objects are referred to in different kinds of formulae for specifying identifier relationships, being input and output of protocols for establishing identifier and identity relationships, requesting data, as well as specifying data to be released, and any formulae derived from the latter. The differences of those uses of identifier objects are explained in Sec. 5.

Example 4.5 (Formula over identifier object) *An example of a formula fragment over an identifier object is given next, with the holder and subject being the same term, thus referring to the same party.*

$$\dots p.\text{holder} = \text{"user4567"} \wedge p.\text{subject} = \text{"user4567"} \wedge \\ \text{Eq}(p.\text{subjectId}, \text{"user4321"}) \dots$$

Registration identifier objects. When a party registers in a system as discussed in Chapter 3, it first creates a registration identifier object on which a registration identity can be issued. The specificity of a registration identifier object is that it can be established without any preexisting relationship with the other party. In our credential protocols, a registration identifier object corresponds to the generation of a new private key of the party. Any other identifier object is established based on an existing identifier object, that is, using the same private key.

Domain identifier objects. A *domain identifier object* is a special kind of identifier object with the meaning that a party may only establish a single such object with one other party comprising the same *domain identifier*. This is a well-known concept in privacy-enhanced identity management. The domain identifier can be freely specified by the party the identifier is established with and typically delimits different scopes of this party. A domain identifier object has an additional special attribute *domain* :: str and realizes the concept of a *domain pseudonym*.

Example 4.6 (Domain identifier object) *For example, in an e-learning service, the service provider may require that users can register only under one identifier*

per course and thus require a domain identifier to be used:

$$\dots p.\text{holder} = \text{"user4567"} \wedge p.\text{subject} = \text{"user4567"} \wedge \\ \text{Eq}(p.\text{subjectId}, \text{"user4321"}) \wedge \text{Eq}(p.\text{domain}, \text{"Introduction.to.Economics"}) \dots$$

The use of such identifier objects allows a service provider to restrict users to a single registration for a service. A cryptographic pseudonym system can enforce the uniqueness of such identifier objects for each user (private key) and domain string. When proving holdership of a domain identifier object, its domain needs to be always revealed, otherwise the underlying cryptographic protocol must terminate with failure as the purpose of the domain restriction would be defeated. Not all kinds of identifier objects may also be domain identifier objects.

Domain identifier objects realize the concept of *domain-specific pseudonyms* or *domain pseudonyms*. They are relevant for all practical applications where it needs to be ensured that a party can only establish a single identifier with a service provider. As an example for the area of e-government, federal regulations in Austria stipulate that different domain-specific identifiers, denoted as sector-specific identifiers, be used with different public sector domains.

Public pseudonyms. As introduced in Chapter 3, a public pseudonym is one a party uses with all players in the system. For a party with a public key that is associated with the party through standard PKI-based public key certificates (PKCs), the public pseudonym can be uniquely derived from the PKC. For pseudonymous certifiers, it can be computed from the Fiat-Shamir-signed statement, which conceptually corresponds to a standard public key certificate—it associates attributes of the key holder to the key. Because in our model only names that have been established in some form as pseudonyms can be used with other parties, we define a public pseudonym to be established with everyone or parties that have obtained the certificate the pseudonym is derived from.

4.4.8 Delegation of Attribute Authority

Delegation of the authority of making attribute statements about a party as discussed in the model in Chapter 3 can be realized through *delegation of identities*. We use the approach of delegating identities or parts thereof and indicating for such identities that they are delegated ones. Delegation of an identity is defined as the process of a delegator giving a delegatee the power to use the attributes of the identity, possibly for an agreed constrained set of purposes.

The case of an identity being a *delegated identity* needs to be encoded in the identity which we accomplish through the reserved attribute *delegation* of type `bool` which is set to `true` for delegated identities and false otherwise. Our modeling through an explicit delegation attribute allows for distinguishing the use cases of delegation of an identity and making statements about third parties being the subjects without delegation. The use of the attributes *holder* and *subject* of an identity with and without delegation is discussed in Sec. 4.8.

The holder of a delegated identity is, following our definition of holder, the delegatee while the subject is the delegater. This assumes that the delegater is the subject of the identity it delegates. The attributes *holder* and *subject* must be instantiated accordingly to express those functions of the holder and subject parties. The case of a third party being the subject is possible as well, but not discussed further in this work.

Both a formula describing the identity in an identity relationship and a data statement made to a verifier need to reflect the delegation attribute. In our examples we often omit the delegation attribute, unless we focus on expressing such semantics.

When delegating an identity, the identity type of the delegated identity remains the same as the type of the base identity of the delegation in our model. Conceptually there are use cases that can benefit from delegating a subset of the attribute information of an identity and thereby realize a notion analogous to data minimization towards the delegatee, namely to delegate only what is required for the intended purposes of the delegation. This is, e.g., useful for limiting the damage in case of misuse of the delegated identity by the delegatee or prevent certain cases of misuse in the first place by not giving away authority over certain attributes. Though, we make the simplification of not changing the identity type in a delegation for the reason of the technically simpler handling. The main arising difficulties are that reducing the delegated attribute information of an identity requires additional consideration. It can be addressed, for example, through delegating a subtype of an identity, or any part of the attribute information that is required while not changing the identity type. The first approach requires an elaborate subtype hierarchy to function, while the second approach follows the approach of minimizing the delegated attributes best with the implications of not having all attributes contained in the delegated identity that its type mandates. Thus, the second approach is more suitable in terms of privacy, though, may require issues with the type system to be resolved. In terms of architecture, the second approach is realized by the formula describing the delegated identity making assertions only about the attributes authority of which has been delegated, and not about other attributes comprised in the delegation identity type. This makes the matching process of identity relationships with policies applicable without modifications. A detailed discussion of the solutions for minimizing the delegation of attribute authority in the context of our architecture is left open to future work.

4.4.9 Transfer of Identities

A related concept to the delegation of identities is the *transfer* of identities from one party to another party. With transfer we mean that both the holdership and subject function for an identity are transferred, on request of its current holder and subject, to another party, the future holder. Transfer is realized by invalidating the identity being transferred, the *source identity*, and creating a new transferred identity, the *target identity* of the transfer. The concept of transfer of identities may be important for many use cases in a future electronic society for replacing traditional credentials with electronic ones in different domains of life.

A transfer of identities is meaningful for identities with attributes specifying transferable entitlements assigned to its holder, but not for identities the attributes of which make statements about the civil (legal) identity of the holder. Prominent examples for the further case are event tickets or vouchers which are allowed to be transferred by their holder (owner), much like the corresponding credentials in the physical world. Clearly, identities certifying civil identity attributes must not be transferable as a transfer would change the subject, thereby violating the intention of the issuer and allow for pooling of identities and thus allowing its holder to make false statements and possibly obtain access to resources it would not be able to access otherwise. The issuer needs to decide for each identity type based on the kind of attributes it comprises whether identities of this type may be transferable. This is out of scope of the data modeling formalism and depends on real-world attribute semantics available to the issuer.

Transferable identities allow for realizing many real-world use cases of transferable physical credentials. Examples for such are movie tickets a person buys for a group of friends and then distributes to them, vouchers that can be passed on through a chain of arbitrary length of holders, or a party invitation that can be passed on to a friend. The identities in those use cases have the property that their holder equals their subject as the attributes of such identities always apply to the holder. If the latter were not the case, we would be speaking of a delegation and not a transfer setting. Attributes of transferable identities typically make statements about entitlements of the holder and subject.

There are two main differences to delegation, namely that the transferring party gives away all rights and capabilities associated with the transferred identity, while in the delegation setting the party may retain rights and capabilities or regain them once the delegation has ended, and that the subject in a transfer is changed to the new holder and does not remain the original subject as is the case for a delegation. The subject is handled like this because for a delegation the main purpose is that authority over the attributes of the delegater is (temporarily) given to the delegatee with the intention that the latter make identity statements on behalf of the further.

For a practical system, an implementation of easy-to-use protocols for transferring (low-value) identities, such as cinema tickets, vouchers and similar items, can leverage mobile handsets like smart phones for realizing real-world scenarios which are currently handled with physical credentials as in the examples mentioned above. Thus, the simplicity of real-world credential handling can be transferred to the electronic domain, thereby obtaining practically simple, yet effective, systems handling those everyday use cases for credential management.

From a data modeling perspective, a transferred identity is modeled like any identity. Transferability of an identity needs to be expressed as part of the identity type as metadata outside of the data model.

4.4.10 Certification Metadata

Certification metadata are data associated with an identity, also denoted the *base identity* in this context, and describe aspects related to the certification of this

identity. This includes, as the main aspect, the specification of the certifier, e.g., by specifying it through a single identifying constant term, identifying combination of attributes, or through a non-identifying combination of attributes. In addition, certification metadata comprise the temporal validity of certification and protocol-related parameters, such as the protocol suite the identity is based on.

Certification metadata are required in order to allow a holder of an identity to compute a fulfilling statement for a data request and a recipient to express the certification requirements when requesting data for an authentication of a data statement as well as to make its trust decisions regarding the identity. Without certification metadata being available for identity data, it is typically not possible for a verifier of the identity data to make a decision on whether to trust the data for the purpose at hand. For this reason, those metadata are equally important for making a policy-based access control decision than the identity data itself they are associated with.

4.4.10.1 Temporal Validity and Protocol Identifier

The temporal validity and protocol identifier are modeled as reserved attributes of the identity in a straightforward way, thus also following the approach of tight integration with the identity data and are used following the pattern of (4.4):

$$\begin{aligned} \dots \wedge \text{Eq}(c.\text{validFrom}, d_1) \wedge \text{Eq}(c.\text{validUntil}, d_2) \wedge \\ \text{Eq}(c.\text{protocolSuite}, p) \wedge \dots \end{aligned} \quad (4.4)$$

The attribute *validFrom* is defined to be equal to the constant d_1 of type **date** or one of its granularized subtypes and *validUntil* is defined to be equal to d_2 , also of a date type. The protocol suite which the identity is based on is specified through the attribute *protocolSuite* which is equal to a constant p of type **str**.

4.4.10.2 Certifier Metadata

For modeling its certifier metadata, an identity comprises the attribute *certifier*. The certifier metadata for a base identity c are expressed by associating the identity with its certifier using a term u , a constant term referring to the certifier. This is accomplished through an equality relation between the *certifier* attribute of c and constant term u and, optionally, expressing predicates, accordingly related to u , for specifying the certifier. This can be achieved in either of the following ways or combinations thereof. (a) Specification of the certifier u to be the subject of another identity, the *certifier identity* cid , through the *subject* attribute of the latter, or an identifier object of the certifier, the *certifier identifier object*, in case it is only referred to through an identifier object, or a combination thereof or multiple objects of either kind. In the following, we only present details for the case of using one certifier identity—certifier identifier objects or combinations are handled analogously. The certifier identity cid is—like any identity—defined as a function specifying its attributes and can be described through predicates over its attributes analogous to

any identity. Analogously, the certifier identifier object is specified through its attributes. The statements over a certifier identity cid or certifier identifier object form a sub-formula $\tilde{\phi}$. The formula fragment (4.5) shows the general pattern of expressing the certifier metadata in statement-type formulae. In request-type formulae, this may be generalized as explained in Sec. 4.6.

$$\dots \tilde{\phi} \wedge \text{cid.subject} = u \wedge \dots \wedge \text{c.certifier} = u \dots \quad (4.5)$$

The term u is a term used for referring to the certifier, that is, typically a publicly used term corresponding to a public pseudonym of the certifier that is used to refer to this certifier by every party or a term representing a pseudonym for the case of a pseudonymous certifier. In either case, there is no need to use different terms in different references to the referred-to certifier. The holder does not need to be expressed for a certifier identity or certifier identifier object in a data statement because this identity is used (referred-to) by a party who is not its holder and a statement about the subject is sufficient for achieving the association of the certifier identity, and thus the certifier, with the base identity. (b) Specification of the certifier through property predicates and not referring to an identity or identifier object of the certifier. This approach is suitable for specifying the certifier for an identity in an abstract way in data requests and deriving the set of concrete applicable certifiers through reasoning over a certifier ontology. For this approach, the certifier is referred to through a variable u . The structure for this approach is presented in (4.6).

$$\dots \wedge \text{Property}(u, \text{"MinimumAssuranceLevel"}, \text{"high"}) \wedge \dots \wedge \text{c.certifier} = u \dots \quad (4.6)$$

Note that the terms ‘certifier identity’ and ‘base identity’ used above are contextual and meaningful only in the context of a formula, as both refer to, by definition, identities. A *base identity* is an identity that has a subject that is not the certifier of another identity in the formula. A *certifier identity* is an identity the subject of which is the certifier of at least one other base or certifier identity. Without considering the context of a formula, a certifier identity is a regular identity of the certifier with the intention that it be used by other parties in relation to identities of them.

The user attributes for characterizing a certifier identity should be aligned with what the underlying technology, which is PKI technology in the standard case of leveraging today’s public key infrastructures, supports for establishing integrity of the statements about the identity.

Example 4.7 (Certifier metadata) *The following example expresses that the party referred to by term u is the certifier of dl and that u is as well the subject of the identity cid , the certifier identity. Via the identity cid , the statements about the certifier u are made, in the example only the statement that the value of its attribute *uniqueId* equals “German_eDL_Issuer”.*

$$\dots \wedge \text{Eq}(\text{cid.subject}, u) \wedge \text{Eq}(\text{cid.uniqueId}, \text{"German_eDL_Issuer"}) \wedge \dots \\ \dots \wedge \text{Eq}(\text{dl.certifier}, u) \dots$$

The example uses the term `cid` for referring to the identity of the certifier `u` and specifies the attribute `uniqueId` of the certifier to be equal to “German_eDL_Issuer”, building on the assumption that this attribute can uniquely identify parties in the system through a string. For X.509 certificates specifying attributes of a certifier, the uniquely identifying name is the distinguished name presented in the certificate. Clearly, more complex sub-formulae as in the example can be used for specifying the certifier’s identity, including the use of disjunctions in data requests.

Our approach for specifying certifier metadata makes the language expressive in terms of being able to refer to any of a set of certifiers. This is crucial for expressing certification requirements in a data request being part of an access control policy. Using this approach is also very dynamic as a policy author is free to refer to any attributes of certifier identities in the specification of the certification requirements as well as using ontology-based generalizations.

We argue that the idea of specifying a certifier by once again reverting to the concept of identities is a natural choice. We cannot see a strong reason for introducing an additional concept for modeling the data of parties acting in the role of a certifier and thereby complicating the language. Particularly, identities are issued to certifiers in a way that is conceptually the same as identities being issued to other parties: Attributes of which they are the subject are certified as part of an identity. In addition, for a certifier, the attributes also include a public key, thereby resembling a public key certificate. Thus, each party is described via identities in our model regardless of what kind of party it is, that is, under which combination of the roles data provider (user), service provider, certifier, or other roles it acts. There is no conceptual or modeling difference between identities with the subject being a certifier or identities with the subject being any other kind of party, e.g., a user. This homogeneous modeling of identities contributes to a simple yet powerful data model and resulting system for privacy-enhanced identity management.

Analogously, the conditional data recipient of a conditionally released identity is specified like the certifier of an identity, that is, the discussions here apply analogously. An opaque identity does not need a certifier to be specified as there might be multiple certifiers for different attributes, deducible from the formula through which it is specified.

Anonymous and pseudonymous certifiers. An interesting use case is that of pseudonymous certifiers, e.g., users who may become certifiers in a certain context and issue identities to other users, while being known on the basis of pseudonym or attribute statements rather than unique identifiers. Such use cases take the idea of attribute-based access control one level further towards certifiers being specified through attributes instead of necessarily through unique identifiers. These use cases can be expressed easily in our data model with its standard mechanisms by specifying a certifier through its attributes using the presented approach.

A use case of a party being a pseudonymous certifier can be technically handled as follows: The party creates an SRSA-CL [CL02b] or BL-CL [CL04] or other suitable signature key pair for issuing private certificates and binds a pseudonym

and attribute statement to the key using a Fiat-Shamir signature [FS86] based on a data statement. The data statement comprises pseudonym and attribute information about the party and conceptually corresponds to the certifier identity backed by a public key certificate of an identified certifier. The party can subsequently use the SRSA-CL, BL-CL, or other private key for issuing private certificates to other parties, and other parties can verify proofs based on such certificates based on the corresponding public key. The public key remains bound to the data statement such that a reputation can be established and associated with it over time.

Following this, the case of a pseudonymous certifier can be handled with the same concepts as any other certifier, which shows the generality and flexibility of our architecture. A difference is in the concrete mechanism: the identity of a pseudonymous certifier is endorsed through a Fiat-Shamir signature over a statement created by the party while in the conventional case standard PKI technology is used. That is, the party can, in the Fiat-Shamir case, itself choose the attribute statement and create a proof for it based on pseudonyms and credentials, instead of requiring that a single other party certify the identity as in the PKI case. Conceptually, both cases represent a pseudonym and attribute statement about the holder of the public key the statement is cryptographically associated with.

The special case of a one-time signature key pair created through the above approach technically works in the same way and for each such key pair any party out of the group of parties specified through the data statement could be the holder and subject. A fresh association between key and statement is created for each such key pair using an appropriate statement specifying the key holder. This allows for special-purpose use cases and has as a major drawback that no reputation can be established for the individual issuer, but only for the group defined through the attribute statement. Also, it increases the complexity of the key management as public keys also need to be provided to the recipients of data statements made with credentials issued by the key holder.

Use cases for the approach of pseudonymous certifiers can be found in collaborative applications or communities where pseudonymous users can obtain identities certifying attributes related to them from the collaboration or community platform and, using attributes of those, create a signature key pair with an associated user-chosen attribute statement. For example, attributes could be the member status or competence in certain areas of the community. The user can then assign attributes to other users by issuing certificates much like any issuer can do. Thereby, those users might obtain permissions on the platform by proving holdership of attributes or statements about attributes issued by a party holding certain attributes without regard to the issuer's civil identity. In today's open systems, one can think of a plurality of further use cases where it makes sense that pseudonymous certifiers issue identities to other parties and are not known by their civil identities, but only by attributes showing a status or capabilities the party has. Use cases like this take the idea of attribute-based specification of parties one level further and applies it also to parties acting as certifiers in our identity system.

Self-certified data. Another practically important use case in our architecture is *self-certified data*, also referred to as *declarations*, where the latter term has been established in related work [BS02]. The characteristic of a self-certified identity is that its holder equals its certifier. The terms used for referring to the holder and subject need to be derived accordingly at creation time of the identity as explained in other parts of this thesis. In the data representation, this is handled by associating the *certifier* attribute of the identity with the holder or subject term used for the party.

Discussion. As one specifically interesting feature, our approach of modeling data through attributes of identities and associating certifiers with such identities allows for counter-intuitive and powerful functionality such as expressing predicates that comprise attributes from multiple certifiers, that is, a single predicate can be jointly endorsed by multiple certifiers. Such statements can be proven with private certificate technologies through applying cryptographic protocols accordingly. An example for this is an equality predicate over the *lastName* attribute of two identity relationships of a user, both backed with private certificates. In this example, the whole predicate is stated by the user, with the not-released attribute values being endorsed by different certifiers, leading to an interesting situation of endorsement for the statement. Most other methods for specifying certifiers would not allow for modeling such complex endorsements. This powerful feature of combined endorsement of predicates is only one reason for having chosen our approach of specifying certifiers in formulae.

In a technical realization, a certifier identity is in all currently envisioned scenarios backed either by a standard public key certificate as cryptographic material or a Fiat-Shamir-based public-key certificate based on pseudonym and credential protocols. See Sec. 4.8 for a brief discussion related to modeling of identities based on credential protocols and conventional signatures.

4.4.11 Type System

We next explain the typing scheme underlying our language, comprising typing of the terms of the language through a typed logic as is standard as well as an additional typing mechanism for associating types with objects such as identities.

The types *idfo*, *id*, *crid*, and *oid*, define the terms that can be used for referring to identifier objects, identities, conditionally released identities, and opaque identities, respectively. We do not specify those types here in terms of their extensional definition, but leave this open to a concrete implementation. The type *att* specifies the terms for attribute names and its detailed specification is also left to a concrete system implementation.

Attributes not being any of the *reserved attributes*, e.g., the attributes *holder*, *subject*, *certifier*, *recipient*, *condition*, *subjectId*, *subjectIdWithCertifier*, or *type*, are referred to as *user attributes*. The following *basic data types* are defined for the user attributes, where the set $\{Y, M, D, h, m, s\}$ of date granularities is introduced

further below:

$$\mathcal{T}^{\text{dat}} = \{\text{int}, \text{str}, \text{bool}, \text{date}, \cup \{\text{date}^\xi : \xi \in \{\text{Y}, \text{M}, \text{D}, \text{h}, \text{m}, \text{s}\}\}\} .$$

This set of basic data types can be extended with additional types and corresponding relation predicates the signatures of which comprise arguments of those types in the future. Such type extensions require also according additions in the semantics of the language and its implementation through cryptographic protocols. We next present details on the defined data types and explain the choices we have made for the type definitions. Terms of the data types in \mathcal{T}^{dat} are self-referential, which means that they are always interpreted with themselves.

The reserved attributes *condition* and *type* have values of type $\text{str} \in \mathcal{T}^{\text{dat}}$, and *subjectId* and *subjectIdWithCertifier* of type idf . The reserved attributes *holder*, *subject*, *certifier*, and *recipient* have values of type pty and attribute names are of type att . The type pty refers to parties in the system and different terms can refer to the same party in an interpretation, which is crucial for the semantics and discussed in great detail later. The type idf refers to identifiers of identifier objects expressed through the *subjectId* attribute or identifiers of identities expressed through the *subjectIdWithCertifier* attribute—terms of this type are self referential in an interpretation like basic types.

Types are denoted through the operator $::$, which we use for both the built-in types of the logic as well as the static identity types for associating a language element with a type. A constant or variable can have multiple types, which is important for handling date-typed elements. Proper typing according to our typing scheme is a prerequisite for the well-formedness of formulae in our language. We refer the reader to related work [CMN⁺10] for a formal definition of a type derivation calculus for a related language for specifying attribute requirements of parties for privacy-preserving access control. For typing of identities, we define further typing mechanisms as explained in Sec. 4.4.11.2 that enhance the expressiveness of our language compared to the one of related work [CMN⁺10].

Types are stored and communicated with formulae in any implementation of our system, though we often do not make the types explicit in this work for reasons of notational conciseness when they are clear from, or not of primary interest in, the context of the discussions.

4.4.11.1 Typing of Attributes

Attributes are assigned types through sorted logic as discussed further above and as is standard. We permit subtyping for attribute types in order to make comparing attribute values through predicates in such subtype hierarchy possible. An example for this are our multiple data types for realizing date types of different granularities. When expressing a predicate on two attributes or an attribute and a constant, this can be done depending on the data types irrespective of the types of the identities the attributes are part of—all identities are of the most abstract identity type id . This approach is in line with the goal of having high expressivity and allowing for uses as discussed above for the identity types in this section.

Let a be an attribute with the type of its attribute value equaling β . For our language we assume that each attribute name a is, in a concrete system, always associated with a single type β for all of its uses in identities or other objects of different types. This requires using a naming scheme that allows for world-wide unique identifiers, though, allows for an open definition of attributes. The URI scheme [BLFM05] of the World Wide Web Consortium (W3C) is the best practical example of a naming scheme achieving this. A given attribute has one of the basic data types of the set \mathcal{T}^{dat} , the type **pty**, or the type **idf** as its type.

Basic data type system. For a system using our logic-based data representation, we assume a fixed *basic data type system* \mathcal{T} to be available. \mathcal{T} defines the basic data types $\mathcal{T}^{\text{dat}} = \{\text{int}, \text{str}, \text{bool}, \text{date}\} \cup \{\text{date}^\xi : \xi \in \{Y, M, D, h, m, s\}\}$, function and relation symbols defined on those types, and the corresponding interpretations.

The equality predicate $\text{Eq}_{\beta \times \beta}$ is available for any type $\beta \in \mathcal{T}^{\text{dat}}$ of our basic data types. The negation thereof, the non-equality relation $\text{Neq}_{\beta \times \beta}$, is also available for all basic data types $\beta \in \mathcal{T}^{\text{dat}}$. For totally ordered types, also relations on the ordering are available for better expressiveness. We support the inequality predicates $\text{Lt}_{\beta \times \beta}$, $\text{Leq}_{\beta \times \beta}$, $\text{Geq}_{\beta \times \beta}$, $\text{Gt}_{\beta \times \beta}$ for the type $\beta \in \{\text{int}\}$ and partially for $\beta \in \{\text{date}\} \cup \{\text{date}^\xi : \xi \in \{Y, M, D, h, m, s\}\}$. The basic data type system determines the interpretation of ground terms of the basic data types for every interpretation \mathcal{I} for our logic. Thereby, it defines those aspects of the system that are related to the definition of its basic data types, thus aspects common to all parties and that require consideration already in the system implementation. Further aspects, e.g., the typing of identities, are reflected in a more flexible way through additional typing mechanisms.¹⁹ \mathcal{T} is known by all parties in the system and required for implementing the basic data type semantics in the automated deduction procedures and interpreting formulae expressed in the logic.

Composed terms. We are only interested in formulae which are well-typed with respect to our type system. Constants and variables have their type always “built in” following the standard approach of sorted logic—cryptographic objects only the respective generic type from the set $\mathcal{O}^{\text{obj}} = \{\text{idfo}, \text{id}, \text{crid}, \text{oid}\}$ corresponding to it.

For composed expressions, referred to as *attribute references*, for constants or variables c referring to cryptographic objects, we have that $c.a :: \beta$ if and only if the attribute a is associated with type β . Such definition does not capture the subtyping of c as defined through the static subtyping system of Sec. 4.4.11.2 below and it may happen that for a formula, although well-typed according to the sorted logic, $c.a$ may be undefined because of the attribute a not being part of the object type for c defined through static subtyping. Such formula is ill-typed in our extended type system, although being well-typed in the sorted logic.

For typing the attribute reference $c.a$ while ensuring that ill-typing cannot happen, we define the typing of such expressions based on the type of the object, e.g.,

¹⁹In a related approach [CMN⁺10], identity types are captured by the corresponding type system already.

identity, and thereby leave the standard typed-logic approach for typing: We define that $c.a :: \beta$ if and only if there exists an object type τ such that a predicate $\text{Eq}(c.type, \tau)$ is part of the top-level \wedge -node²⁰ of the formula comprising the attribute reference to be typed or a predicate $\text{Type}(c.type, \tau)$ can be deduced from the formula, $a \in \mathcal{A}_\tau$, and the type associated with a is β . Evaluating the type following this definition requires the considered formula to express an Eq predicate over the attribute *type* of c and a constant in its top-level \wedge -node, otherwise the type of c is undefined.²¹

For a constant or variable term C without its *type* being specified through a predicate in the considered formula, its possible types can be deduced through the referred-to attributes in the formula and their respective types with the effect that multiple types may apply to C . This is a desired property when expressing a data request because of keeping the applicable identity types open and governed only through the referred-to attributes. This is referred to as dynamic typing and introduced further below. The logic-defined type of C is always *id* or one of the other types for cryptographic objects.

For arithmetic expressions composed through the application of operators (functions), we have the following recursive definition, where DefOps is the set of operators defined for this type through \mathcal{T} :²²

$$t_1 \circ t_2 :: \beta_1 \text{ iff } t_1 :: \beta_1, t_2 :: \beta_2, \circ \in \text{DefOps} . \quad (4.7)$$

Integer. The integer type *int* comprises the integers or, for formal reasons, an interval over the integers which is sufficiently large for our practical purposes. In a technical implementation using a credential system as done in this thesis, the type is always constrained to a finite interval $[-2^{l_m} - 1; 2^{l_m} - 1]$ over the integers. This interval is specified through a length parameter l_m , resulting in a 0-centered interval over the integers for a concrete system, where the maximum length of elements of the interval in binary representation is typically in the range of multiple hundred bits. The type definition for *int* abstracts from those size limitations of integers and uses the group of all integers as the set of permitted elements which is a practical approach as integer attributes used in practical applications are typically contained in the interval of length l_m , for an appropriate l_m . The finite integer interval is also known as and henceforth also denoted as *integer interval*.

Date. Attributes resembling dates or times are prominently used in the identity management context, e.g., to express dates of birth used for age verification, or the expiration dates and times of certified identities. We henceforth refer to date and

²⁰With top-level \wedge -node of a formula ϕ we mean the root connective, which is conjunction, and its successor predicates in a prefix notation of ϕ with respect to the connectives.

²¹For the case of the predicate over the *type*-attribute being located within an \vee -branch of the formula, the situation is more complicated as the object's type would hold only for this branch. This situation is neither supported nor relevant for cryptographic protocols based on the private certificate technology and the corresponding formulae.

²²Note that for date-typed operands, the defined operators are defined over one operand specifying a date and one integer operand specifying a duration.

time also simply as date. We define multiple date types for expressing the date and the time component using different *granularities* of the date and time.

The syntax for a date value is such that it comprises a sequence of *components* and is represented starting with the most coarse-granular component, going to the finest, each finer-granular component being appended to the right and having a predefined number of digits. We express, for example, a date in day-granularity in the format YYYY-MM-DD, where YYYY are 4 digits for the year, MM two digits for the month, and DD two digits for the day, using the ISO date format [ISO04] with separating “-”-symbols. When also considering the time of the day, the time is indicated in 24-hour format analogously as follows and appended at the right to the date: *hh-mm-ss* with *hh* being the hours, *mm* the minutes, and *ss* the seconds. Using such granularized date types increases the expressivity of the system. A concrete system instance can also be deployed with a simplified handling of date types, e.g., only a single type, if this is considered sufficient.

Finer-granular types with accuracy below seconds can be represented in a straightforward way, though, are not further discussed in this thesis because the definition and handling is analogous to what is presented. Such finer granularities are—as far as the validity times of identities are concerned—less relevant for the identity system we have in mind for authentication through private certificate protocols over electronic communication networks, mainly the Internet, where communication latencies are often in the multi-hundred millisecond range already, thereby ruling out the need for a finer granularity for those attributes.

We assume that in our system time values are always represented in UTC time [ITU02] and time zones are considered outside of the system we discuss, e.g., through functionality at higher layers, e.g., in the user interface component or as time offset in policies. This is conceptually clean and simplifies our discussion by avoiding time zone issues without making practically limiting constraining assumptions on the system. Certain aspects related to time zones can be seen to be completely outside of the technical domain of the system, e.g., the question whether an identity that is still valid in its issuing time zone, but invalid in the verifying time zone, shall still be accepted as valid by a verifier.

Let the symbols Y, M, D, h, m, s be denoted as *granularities* and their ordering be defined through a transitive relation $\mathcal{G}^{\text{date}}$. The date types that we define are the *generic type date* and the *granularized types date^ξ* for dates, where $\xi \in \{Y, M, D, h, m, s\}$ indicates the granularity the corresponding date type is expressed in, using the granularity specifiers: Y for years, M for months, D for days, h for hours, m for minutes, and s for seconds. The set $\{Y, M, D, h, m, s\}$ of granularity specifiers is also referred to as $\mathcal{T}^{\text{date}}$ and may be omitted when clear from the context. The type *date* is the generic date type and an element of this type can have any granularity and can be compared with another element of type *date* of any granularity or with an element of any of the granularized date types *date^ξ* with $\xi \in \mathcal{T}^{\text{date}}$. For two arguments of granularized types, those types have to be the same in order that the arguments be comparable, e.g., an argument of type *date^D* can be compared with an argument of type *date^D*, but not with an argument of type *date^h*. In terms of the extensional definition of the types, this means that

$$\llbracket \text{date} \rrbracket = \bigcup_{\xi \in \{Y, M, D, h, m, s\}} \llbracket \text{date}^\xi \rrbracket.$$

Let the transitive relation $\mathcal{G}^{\text{date}}$ be defined such that for the granularities introduced above the following holds: $\{(Y, M), (M, D), (D, h), (h, m), (m, s)\} \subset \mathcal{G}^{\text{date}}$. All elements resulting from the transitivity property of the relation are in addition part of $\mathcal{G}^{\text{date}}$ as well, only not exhaustively listed here.

Date types can be subtyped according to a subtype hierarchy which has the generic type **date** as its root and the granularized date types **date** $^\xi$ as direct successors. An element typed with **date** can take on the value of an element typed with any of the types $\{\text{date}\} \cup \{\text{date}^\xi : \xi \in \{Y, M, D, h, m, s\}\}$, while an element typed with a granularized type **date** $^\xi$ can only take on values from the domain corresponding to this single type.

For identities of a given identity type, the certifier of the identity decides on which data type to use for its date attributes, e.g., validity dates, depending on the use case and issues the attributes in according granularity.

Only additive relations on dates are supported as this is sufficient for most use cases requiring date arguments. For example, this allows us to specify that a reference date against which a check of the expiration date of an identity is to be performed has to be 3 months in the future, which may be interesting for travel-related use cases.

String. For resembling real-world identities in a practical system, string attributes are crucial as they are required for a plethora of natural persons' attributes such as name and address attributes.

A practical system needs to support both length-limited and length-unrestricted strings, the further being interesting for modeling the length-limited attributes of today's physical credentials, the latter may be of importance for representing conditional release conditions that may comprise general terms and conditions of service providers or a facial image attribute of a photo id as those may be substantially larger as standard users' identity attributes.²³ Note that length-unrestricted strings may be strings of limited length which is sufficiently large for all practical purposes in our model for formal reasons. From a modeling perspective, we have the options of exposing a length restriction of one of multiple string types at the data representation language or to only expose one type capturing both length-restricted and unrestricted types.

Option (1). A first option we consider for realizing length-restricted types is to enforce length restrictions for the appropriate types at the level of the type system. Comparison predicates, such as multiple overloaded predicates for string equality need to be defined such that they can be used to compare strings of the different types with each other. Particularly, strings of types with different length

²³We assume for simplicity of the type system that a biometric identifier such as a picture of a face in a government-issued electronic identity (eID) document be represented as a **str**-typed attribute in an identity. Extensions of the type system are possible, though, not crucial for the understanding.

restrictions should be comparable with each other as well as with a generic unrestricted type. This approach may be facilitated by a subtype hierarchy over the types with subtypes of decreasing sizes being subtypes of the generic type.

Option (2). A second option we consider for modeling strings is to define a single type `str` in the type system which does not have any limitations in length. Any such restrictions need to be captured otherwise, such as in the metadata of identities, and enforced in processes such as the protocol for establishing identity relationships and for the definition of the relation predicates. This approach has the advantage of only a single type `str` being exposed and thus there being no need for the hierarchical type system and definition of multiple overloaded predicates operating over the different string types of Option (1). The only noteworthy drawback is the lack of explicit length restrictions of strings in the type system.

For its conceptual simplicity, we have chosen Option (2) of defining a single type `str` of unconstrained-length strings, thereby avoiding the complications of Option (1). In case a certifier needs to impose length restrictions—as is typical for common credentials such as electronic identity (eID) cards or electronic driver’s licenses—it can do so by enforcing such restrictions when setting up an identity relationship, that is, through a certificate issuing protocol. The case of a certifier issuing an identity with an attribute value being that of an opaque attribute it has received is similar for both modeling options in that the certifier needs to trust that the issuing party of the opaque attribute has enforced the length restrictions properly. The difference is that the maximum attribute length is not captured formally in the data modeling language in the chosen approach. Because such use of opaque attributes is anyway based on trust that the other issuing party is honest, the additional requirement of trusting the party to properly enforce the stated length of the attribute is not a further limitation.

Boolean. The type `bool` comprises a domain with the elements `true` and `false` and does not allow for expressions to be specified over this type. Unlike for integers, dates, and strings, its definition is straightforward and does not require further discussion.

Identifier. The type `idf` is used for representing pseudonym identifiers used in identifier objects. Like the basic data types, terms of this type are interpreted with themselves in any interpretation.

Party. A term of type `pty` is used for referring to a party in the system. A main difference to the basic data types is that this type is not self-referential in interpretations. Multiple terms in the language can refer to the same party in an interpretation.

Only the logic equality relation $p_1 = p_2$ can be expressed on arguments p_1 and p_2 of type `pty`. Non-equality for parties is not supported to be expressed in data

statements, because it would be impossible to generally establish such non-equality in a cryptographic proof.

4.4.11.2 Typing of Identities

One specific kind of terms are those referring to identities, typed through sorted logic like any other term as explained above. Identity terms and variables may, but need not, have an *identity type*—in addition to the “built-in” type assigned through the sorted logic—associated with them. This identity type defines the identity in terms of its attributes, the data types of the attributes, and technical considerations necessary to execute protocols associated with the identity. The type is realized in the language as an attribute *type* of type `str` of the identity.

An identity referred to in a formula specifying an identity relationship always has such type, while an identity (variable) referred to in other formulae, e.g., in a data request, may not have such type associated for reasons of greater expressiveness in specifying the request. The reason for having this type optional is to respond to requirements of real-world identity management systems: A prominent use case is to request attributes without imposing restrictions on the type of the identity the attribute is comprised in, and defining the identity through imposing constraints on the certification metadata for the attribute. This is an important use case when a party requests values of attributes that need not be certified by a third party and where the type of the identity they are contained in does not matter, or when they need to be certified, though, regardless of the type of identity it is contained in. We allow for such requests to be expressed in a succinct way through the identity concept by not requiring the type of an identity to be always specified.

Example 4.8 (Age verification) *Take as a practical example hereto the use case of age verification which is a legal requirement in many jurisdictions for access to certain services or purchasing certain restricted, e.g., dangerous, goods. For this use case, a predicate over the date of birth attribute is requested, with a plethora of identities being eligible as a basis for constructing and proving such predicate to be valid, and a given minimum strength of certification being required.*

The general structure of this and many other practical use cases based on requiring the authentication of an attribute regardless of the credential it is comprised in benefits from our approach to typing for succinctly expressing a data request for such use cases without enumerating all eligible identity types or using statically defined subtyping hierarchies.

Note that the definition of the type of an attribute of an identity or other cryptographic object further above is non-standard due to the identity or other object type not being expressed directly as a “built-in” type of the logic, for which reason also the processing for type checking of a formula is non-standard.

The notation $\llbracket \beta \rrbracket$ is used in this work to explicitly refer to the extensional type definition of the type β , that is, the set of its elements. For example, a variable $t :: \text{int}$ has type `int` associated, which has the set of the integers as its extensional definition.

A generic identity term $c :: \text{id}$, that is, a term typed with id , is a function symbol for a function

$$c :: [\text{att}] \rightarrow \bigcup_{\beta_i \in \{\beta_1, \dots, \beta_k\}} [\beta_i], \quad (4.8)$$

with $\{[\beta_i] : 1 \leq i \leq k\}$ being the data types related to the attributes of c . That is, an identity term refers to a function from the set of attribute names to the union of the sets corresponding to its attribute types.

The type β of a conditionally released identity or opaque identity is determined implicitly through the attributes of it that are referred to throughout the single formula through which the object is defined. The alternative approach of specifying the type of such object explicitly is cleaner in terms of typing, though, requires that the type be defined and distributed in an integrity-protected manner like for identity types. This reduces the flexibility of specifying policies and complicates the system for handling the additional types.

Static type system. The *type*, or *static type*, of an identity is specified through the *type*-attribute, which we let, w.l.o.g., in the following be attribute a_T . The static type specifies the attributes with their types that identities of this type comprise, expressed as a set of attributes with their types, and is referred to by its type identifier which is a constant term of the language.

Formally, an identity type τ is defined through the attributes $\mathcal{A}_\tau = \{a_1, \dots, a_k\}$ of the identity with their respective types and the reserved attribute a_T having value τ to correspond to a set of functions c , each corresponding to one identity of the identity type τ :

$$[\tau] = \{c :: \mathcal{A}_\tau \rightarrow \bigcup_{\beta_i \in \{\beta_1, \dots, \beta_k\}} [\beta_i]\}$$

For this, it needs to hold that the relation Eq holds between $c.a_T$ and τ and that $c.a_i$ is in the set $[\beta_i]$ for $1 \leq i \leq k$. This definition of the type extension of an identity type does not take subtyping of identities into consideration as discussed below.

Note the difference to the approach of related work [CMN⁺10] where each identity type, referred to as *card type*, is defined through its attributes a_1, \dots, a_k and their respective types β_1, \dots, β_k only—see the extension of a card type in their work for their definition of the typing. In our scheme it is possible to have multiple different identity types comprising the same attributes and their types. This is necessary in a practical open identity management system for allowing different parties to create differently named types with the same attributes and have different parties issue those types. Assuming that not every certifier defines their own attribute types for commonly used attributes, but standard attributes are reused, our approach becomes even more important as the probability of structurally identical identity types increases. Without our generalized view one would essentially require a single responsible authority managing all identity types in the system, which is too strong an assumption for an open system.

Static subtyping. A type hierarchy is induced by relating identity types through their type identifiers in a type hierarchy. The inheritance graph is specified through the identity types being its vertices and each subtype specification (τ_2, τ_1) a directed edge, thereby implementing an *is a* relation between τ_2 and τ_1 . We use the relation symbol $\leq_{\mathcal{O}^{\text{tp}}}$ for referring to the subtype relation, where \mathcal{O}^{tp} refers to a subtype ontology as introduced later. This relation symbol is not part of the data representation language and only used for the discussions related to it.

A type τ_2 being a *subtype* of τ_1 means that τ_2 comprises all attributes of τ_1 with the same types (or subtypes thereof) and possibly additional attributes, thus for types $\tau_2 \leq_{\mathcal{O}^{\text{tp}}} \tau_1$, their respective attribute sets \mathcal{A}_{τ_1} and \mathcal{A}_{τ_2} are subsets of each other:

$$\mathcal{A}_{\tau_1} \subseteq \mathcal{A}_{\tau_2} .$$

The type extension $\llbracket \tau \rrbracket^{\mathcal{O}^{\text{tp}}}$ of an identity type τ with respect to the subtype ontology \mathcal{O}^{tp} is defined as the union over the type extensions $\llbracket \tau_i \rrbracket$ where the types τ_i are τ and all its (direct and transitive) subtypes according to the subtyping ontology \mathcal{O}^{tp} :

$$\llbracket \tau \rrbracket^{\mathcal{O}^{\text{tp}}} = \llbracket \tau \rrbracket \cup \bigcup_{\tau_i : \tau_i \leq_{\mathcal{O}^{\text{tp}}} \tau} \llbracket \tau_i \rrbracket .$$

We use single inheritance as it is sufficiently powerful for the requirements we have in mind for our architecture and it is conceptually cleaner than multiple inheritance while avoiding its complications. Our approach to subtyping realizes key aspects of the standard notion of inheritance, e.g., as used in object-oriented programming languages.

In our model, the type hierarchy for identities is specified through a *subtype ontology* \mathcal{O}^{tp} . The ontology captures the static subtype specifications relevant in a given context in a formal way such that they can be machine processed. Details about subtype ontologies are presented in Sec. 4.10.1.

For realizing the subtyping hierarchy, we define a predicate $\text{Type}_{\text{id} \times \text{str}}$ for specifying types of identities and a predicate $\text{Subtype}_{\text{str} \times \text{str}}$ for expressing the subtype relations where $\text{Subtype}_{\text{str} \times \text{str}}(t', t)$ means that t' is a subtype of t . Based on the static type of an identity as assigned to it through its *type* attribute, the identity takes on all types upwards a given type hierarchy induced through \mathcal{O}^{tp} in addition to its static type. Any of those types can be used at places where one of the super-types is required. The types an identity takes on are reflected through the predicate Type .

The following rules in (4.9) and (4.10) implement the subtyping of identities in the deduction system of our logic, where $C :: \text{id}$ is a variable referring to an identity and $T' :: \text{type}$ and $T :: \text{type}$ are variables referring to concrete types:

$$\forall C, T : \text{Eq}(C.\text{type}, T) \rightarrow \text{Type}(C, T) \quad (4.9)$$

$$\forall C, T', T : \text{Type}(C, T') \wedge \text{Subtype}(T', T) \rightarrow \text{Type}(C, T) \quad (4.10)$$

This rule and a subtyping ontology \mathcal{O}^{tp} needs to be part of the theory \mathfrak{L} used for making derivations in our logic when subtyping is to be considered—see Sec. 4.9 for details on this. Leveraging subtyping for a data request requires that the type of identities be specified using the $\text{Type}_{\text{id},\text{str}}$ predicate instead of the *type*-attribute.

It is important for our language that we do not follow the approach of assigning a concrete identity type to an identity through the means of typed logic as done in related work [CMN⁺10], but rather assign each identity only the generic type *id* in the logic. Also, the subtype hierarchy is handled in a more flexible way as in this item of related work. The approach of using typed logic for typing identities would prevent us from realizing important uses for our language which allow for elegant and powerful ways of expressing data requests and data statements. Particularly, this would rule out the dynamic typing and its expressive power as introduced below.

The interacting parties in a protocol for releasing data need to use compatible subtype ontologies in order to obtain interoperability in terms of employing a reasoning system that allows for making derivations in a mutually understood way.

Architecturally, the subtype ontology \mathcal{O}^{tp} used by a party is based on at least one (partial) subtype ontology, issued and distributed through parties trusted for this purpose. The integrity of the ontologies needs to be ensured, because integrity of the type system is crucial for security of the overall identity management system. A rogue type ontology used by a service provider can, for example, allow an attacker to gain access to resources of this service provider with identities that would otherwise not give it access.

The subtype ontologies used to compose \mathcal{O}^{tp} can be issued by parties acting as ontology providers. This role can be held jointly with other roles, though, trustworthiness of the party for this purpose must be ensured. The support of subtype ontologies from different providers is important for an open system, see Sec. 4.10 for further details.

The issuers of identities of a given type need to adhere to the discussed definition of types. It is crucial that all identities that certifiers issue for a given type comprise the same set of attribute types. If this is violated, identities with different sets of attributes will not be usable for fulfilling certain policies, thus violating the type system as well as the expected system behaviour. Though, it would not necessarily have detrimental effects on security, but rather the availability property of the system would be compromised.

The (static) subtyping of identity types allows for referring to an identity variable of a specific type τ in a data request and fulfilling this request with a data statement referring to a subtype of the policy-requested type τ . This leads to a substantial increase of the expressive power of a policy language in terms of specifying sets of possible identities through those subtypes compared to one without subtyping, analogous to using subtyping through sorted logic, though, with a more open approach to the subtype definition through subtype ontologies.

Taking our identity typing model further, one could allow for different types for one identity in different parts of a formula comprising disjunctions, also extending to subtyping. Certain identity management protocols may be able to take advantage of this, e.g., such based on online certifiers that make identity statements based on

certified statements of other certifiers and can thereby relate different identities in a weak trust model. For credential protocols, though, the language must be further constrained and require an identity type to be fixed in the top-level \wedge -node of a formula.

Dynamic type system. In addition to the (static) type system explained above, our definition of the language allows for implementing a *dynamic type system* which proves beneficial in the context of reasoning in our logic. The dynamic type system allows for an identity specified through a formula ϕ to have a dynamic type τ defined through its attributes that are referred in the formula. It furthermore allows an identity constant or variable used in ϕ to be of a *dynamic subtype* of another identity constant or variable referred to in ψ , where ψ may be ϕ , merely by ϕ referring to at least the attributes ψ refers to in making statements about the considered identity.²⁴ We denote an identity term or variable C_1 being of a dynamic subtype of an identity term or variable C_2 , with $C_1 \leq_{\text{dyn}} C_2$. A dynamic (sub)type is orthogonal to the static types and subtyping of the identities specified through the *type* attributes and \mathcal{O}^{tp} . A dynamic subtype relation $C_1 \leq_{\text{dyn}} C_2$ can hold only if no type attribute or predicate is specified for term C_2 in the corresponding formula, that is, only its attribute references determine the dynamic type, because otherwise the *type*-attribute or predicate would immediately constrain typing to the static scheme and rule out dynamic typing, thereby reducing expressiveness. Exactly for this reason it is not possible to use the approach of subtyping over the identity types through typed logic when the benefits of dynamic typing are to be leveraged. The definition of the dynamic type of an identity generalizes to being defined through the attributes referred to in a conjunction of formulae $\bigwedge_{i=1}^k \phi_i$.

Let the set of typed attributes $\mathcal{A}_{\hat{\tau}} = \{a_1, \dots, a_k\}$ of an identity instance or variable $C :: \text{id}$ be referred to in a formula ϕ , or conjunction of formulae. Let furthermore the attribute *type* not be part of the set $\mathcal{A}_{\hat{\tau}}$. Then, the extension $\llbracket \hat{\tau} \rrbracket$ of the dynamic type $\hat{\tau}$ comprises all functions representing identities of identity types τ_i which comprise at least the attributes of the set $\mathcal{A}_{\hat{\tau}}$ with the according types:

$$\llbracket \hat{\tau} \rrbracket = \bigcup_{\{\tau_i : \mathcal{A}_{\tau_i} \supseteq \mathcal{A}_{\hat{\tau}}\}} \llbracket \tau_i \rrbracket. \quad (4.11)$$

That is, the extensional definition of the dynamic type comprises all identities that match the specification of the dynamic type based on the referred-to attributes with their types referred to in the formula.

The prominent use case for dynamic subtyping is to use any identity of an identity relationship with static type τ with attribute set \mathcal{A}_{τ} such that $\mathcal{A}_{\tau} \supseteq \mathcal{A}_{\hat{\tau}}$ in place of an identity variable of dynamic type $\hat{\tau}$ with attribute set $\mathcal{A}_{\hat{\tau}}$. In this case, it holds that $\tau \leq_{\text{dyn}} \hat{\tau}$ which is a prerequisite for the identity of type τ of a data statement matching the identity of dynamic type $\hat{\tau}$ in the request. Dynamic types

²⁴For simplicity we do not discuss the complications arising from the most general view where the type may be different in different \vee -branches of the formula. Though, this general view of dynamic subtyping is supported by our language and the underlying deductive system.

apply to both constants and variables of identities, conditionally released identities, and opaque identities.

In case a dynamic type of an identity of a data request corresponds to the empty set of identities, this is not a problem in terms of typing. The main implication is that such identity in the data request then cannot be matched by any identity in the system.

The dynamic typing and subtyping is, for instance, beneficial in the context of the matching of formulae ϕ_1, \dots, ϕ_k of a party's portfolio with a formula ψ representing a data request or determining whether a statement expressed as formula fulfills a data request. This is technically realized implicitly through automated reasoning in our logic as presented in Sec. 4.9.2. Capturing the intuition behind this, a formula ϕ_i , e.g., describing an identity, can match parts of a request formula ψ if the attributes and their types and values match with those referred to in the request formula, with the variable of the request being instantiated with the corresponding instance from ϕ_i in the response. Considering this, the dynamic type system is orthogonal to the “built-in” types of the logic.

The following examples sketch the classes of scenarios where dynamic (sub)typing is beneficial and also the relevance of dynamic (sub)typing in practice for the matching of data formulae representing the portfolio of a party against a request. Consider a data request asking for an attribute of an identity without specifying any requirements on the certifier and type of the identity. This reflects the common case of a self-stated (declared) attribute being provided as is used in almost all interactions in today's Web through form filling. The commonality of scenarios for which dynamic (sub)typing is beneficial is that the certifier of identities can be specified in a generic manner, independent of the type of the identities involved.

Example 4.9 (Dynamic subtyping for request predicate) *A concrete example request by a service provider comprises the predicate $\text{Eq}(\text{C.lastName}, \text{L})$ as the attribute request part while not expressing constraints on the certifier or type of C. A user can fulfill this part of the request by using any of its identities of its identity relationships, also such not endorsed by a third party, comprising the attribute `lastName` by revealing the attribute value.*

Example 4.10 (Dynamic subtyping and ontology-based abstractions) *Another example in this domain is the request of an attribute a of identity variable C or a predicate expressed over it and the certifier being specified through the certifier identity `Cid` which is further specified in an abstract manner, e.g., through expressing property predicates based on ontology concepts on it, e.g., for specifying that the `securityLevel` of the certification of this identity be greater than or equal to 3. Regardless of the actual type of the identity C, any identity that fulfills those requirement can be used for fulfilling the request.*

Again, static subtyping as used in this work or in related works [CMN⁺10] would not allow for such generic specification of a data request and thus severely constrain the expressivity of the language.

The above discussions and examples show why static typing for identities is insufficient for realizing certain classes of practically important use cases: A static type system is fixed and does not necessarily align, i.e., may be too restrictive, with the requirements of a policy author for a specific situation. That is, a policy can, through referring to ontology concepts for specifying the certifiers in an attribute-based way and through identity properties, be far more powerful than a fixed subtyping hierarchy over identities. Thus, our work takes a more expressive approach compared to notable related work [CMN⁺10] in terms of simplifying and increasing the expressivity of specifying policies by using a non-standard approach for typing. We find support for such functionality of importance in open systems where data recipients might want to state data requests in a generic and abstract way without policies being repetitive or requiring one to specify applicable certifiers exhaustively.

In a data request, the type attribute may be unspecified for an identity in order to use dynamic typing. Not specifying it in a data request means that any identity with matching attributes can go in place of the requested identity. For a data statement to be proven using credential protocols, the type must be specified for each identity as the cryptographic key is determined through the type. For other technologies for ensuring attribute integrity, the language could be extended to not require that the type be specified for an identity. Though, this additional expressivity is not worth the increased complexity and freedom in the language.

4.4.11.3 Discussion

The combination of our orthogonal type systems results in a flexible overall approach to typing for identities, conditionally released identities, and opaque identities, combining advantages of statically typed and untyped languages in terms of typing of objects. Particularly, it allows for fulfilling real-world requirements of stating a request for attributes without giving information on the type of the referred-to identity. The related language of Camenisch et al. [CMN⁺10] has been designed for similar purposes as ours and follows an approach of typing and subtyping of identities through sorted logic with the inherent drawbacks, though resulting in a simpler typing model and the advantages of using only typing through sorted logic.

4.4.12 Expressions

Expressions are used for realizing identity statements in which multiple attributes of objects are algebraically related with each other. An expression is an arithmetic expression over attributes of objects and constants of types with totally ordered domains and can be an argument of a predicate that accepts an argument of the type of the expression. Expressions are supported for the type `int` in a general form and for the types `date` and `dateξ,ξ` $\in \mathcal{T}^{\text{date}}$ in a simplified form, following requirements of use cases. For `str`-typed elements we do not support expressions for the reason that those would not be practical in terms of efficiency when being realized with credential systems, our focal technology.

Integer expressions. Expressions for int-typed elements have the form of being l products of attribute references and constants and a constant connected through the operator $\circ \in \{+, -\}$, where $+$ and $-$ are the addition and subtraction operators on the integers, and \cdot is multiplication, as presented in (4.12).

$$u_{j_1} \cdot c_{j_1} \cdot a_{k_1} \circ u_{j_2} \cdot c_{j_2} \cdot a_{k_2} \circ \dots \circ u_{j_l} \cdot c_{j_l} \cdot a_{k_l} \circ u_{j_{l+1}} \quad (4.12)$$

Examples for the use of integer expressions and relations other than equality between attributes are use cases such as proving that the monthly total income certified through salary statement identities c_1 and c_2 of a person from its two part-time jobs exceeds a specified amount, as shown in the formula fragment (4.13).

$$\dots \text{Geq}(c1.amount + c2.amount, 8000) \dots \quad (4.13)$$

An example for multiplicative constants unequal to 1 is weighing of reputation scores from different reputation providers who assess a party or other system entity and issue reputation credentials to the party, where weighing is based on the credibility or reputation of the providers, and where a minimum reputation score sum based on multiple reputation identities (credentials) needs to be proven for establishing one's endorsed reputation towards another party.

Date expressions. For dates, the allowed expressions are much more constrained than for the integers. Expressions formed in a more generic way would not add to the functionality, though complicate the system for proving such expressions considerably due to the additional complications resulting from the date types with different granularities. Thus, an expression can be either of the following constructs: a date constant, a date attribute reference, a date attribute connected through $\circ \in \{+, -\}$ with a constant specifying a constant date duration, or a date constant connected through $\circ \in \{+, -\}$ with a constant date duration as shown in (4.14), where u represents a date constant, $c.a$ a date attribute, each of a date type, and d an integer specifying a date duration in the respective granularity. For technical reasons we only allow dates or date durations of the same granularity in one expression on dates. This avoids implicit type conversions which are not generally feasible to be computed in a data-minimizing way when the reference date is not revealed, due to concepts such as leap years and seconds.

$$\begin{aligned} &u \\ &c.a \\ &c.a \circ d \\ &u \circ d \end{aligned} \quad (4.14)$$

Although int-typed date durations are used as arguments in expressions besides arguments of a date type, the type of an expression is always one of the date types.

The above-presented date expressions are—in our view—sufficient for handling the requirements for the use of dates in our system. Expressions with an attribute reference and an integer duration constant are used for comparing date attributes

while ensuring that they be a certain time duration apart from each other, e.g., for expressing that a passport is valid for an additional 30 days compared to another identity. Simple expiration date checks require only expressing an inequality relation between an attribute reference and a constant date or an expression over two constants if an offset to today's date is specified.

Expression example. Next, we present an example of a data statement making use of both integer and date expressions.

Example 4.11 (Expressions) *A concrete example use case for expressions evolves from the area of residence permits in Switzerland. Swiss authorities require, as one option for an EU or EFTA national to obtain a residence permit, to prove one owns sufficient funds. In the current process, this is implemented by the applicant providing one or more account statements certified by the issuing bank to the authorities, thereby also revealing, in addition to the account balance, information such as the account number and possibly transaction records.*

*A privacy-preserving proof for a person fulfilling such requirement of ownership of a minimum amount of funds can be realized by proving a **Geq** predicate on an expression comprising the sum of the balances of multiple bank account identities and the reference constant determined by the authorities. Also, it needs to be ensured that the different account identities referred to indeed represent different accounts, e.g., by expressing non-equality predicates over the account number or serial number attributes of the identities. How exactly non-equality of the identities is expressed relates to details of the definition of the involved credentials and may require predicates over further attributes. In addition, requirements on the freshness of the used bank statements can be expressed through an inequality predicate on the corresponding attributes of one of the types **date** or **date^S** of the bank statements. Certification metadata are omitted as usual in the example fragment of a data statement below and need to, among other things, specify the permitted certifying financial institutions.*

$$\begin{aligned} & \dots \text{Geq}(\text{c1.balance} + \text{c2.balance}, 200000) \wedge \\ & \text{Neq}(\text{c1.accountNumber}, \text{c2.accountNumber}) \wedge \\ & \text{Eq}(\text{c1.currency}, \text{"CHF"}) \wedge \\ & \text{Eq}(\text{c2.currency}, \text{"CHF"}) \wedge \\ & \text{Geq}(\text{c1.stmtDate} + 90, \text{currentDateTime}) \wedge \\ & \text{Geq}(\text{c2.stmtDate} + 90, \text{currentDateTime}) \dots \end{aligned}$$

Example 4.11 explains how to represent expressions on both integers and dates. The integer expression relates two attribute references with constant factors of 1. The date expressions specify an offset to the attribute reference and compare this to today's date for establishing that the statements have a certain maximum age. The *stmtDate* attributes are expressed in day granularity, thus, the integer 90 represents

a number of days. In the practical use case, the party being the holder and subject of those identities can be proven to be the same as the holder of another identity, e.g., an electronic passport, used to reveal further identity attributes for associating the statement on monetary liquidity with the person's civil identity. Thus, the above example is to be seen as an isolated formula fragment which can be embedded into a more comprehensive formula or relate to other formulae through opaque identities established with the use of such.

Note that use cases like this can be expressed through a disjunction of data requests, each specified for a different number k of identities referred to in the expression. Alternatively, the request language could be extended to comprise special provisions for expressing such expression over k attributes of different identities in a generic manner. This could be realized, e.g., through rules expressed in (an extension of) the logic or a macro construct on top of the language.

Overall, use cases employing expressions in predicates are standard, e.g., for specifying that the sum of multiple attributes is greater than or equal to another attribute, for example one of a conditionally released identity, or a constant, or specifying a verification of a date based on a given offset. Similar use cases as the above can be thought of for other areas such as privacy-preserving checking of creditworthiness of a party.

4.4.13 Relation Predicates and Equality

Predicates have been informally introduced already earlier in this section for presenting the language basics. Now we discuss the details regarding the predicates on arguments of the defined data types. The language and its semantics can be extended with further predicates if this is required in the future, possibly also aligned with future extensions of the basic data type system.

Depending on the predicate, each argument can be a generic expression, attribute reference, or constant term of the type matching the predicate signature. No predicates can be expressed explicitly on objects as arguments. The reason for this is that equality over objects can only be expressed implicitly through using the same term in multiple references to the object for expressing that both are the same. Non-equality over objects can be expressed through predicates on suitable attributes, e.g., the unique object identification number or serial number attribute of an identity. We prefer this approach of realizing the semantics of (non-)equality predicates on objects through predicates on their attributes, because allowing for (non-)equality predicates on objects would lead to complications in the mapping of statements to cryptographic protocols while not considerably enriching the expressiveness of the language.

Also, no relation predicates are defined for terms representing attributes of cryptographic objects for the reason that att-typed terms are known to recipient parties of a statement. Thus, predicates over them would not enhance the expressiveness of the language in terms of facilitating privacy or in any other useful way.

4.4.13.1 Integer

For `int` arguments, which are all characterized by comprising totally ordered value domains, we specify the predicates $\text{Eq}_{\text{int} \times \text{int}}$, $\text{Neq}_{\text{int} \times \text{int}}$, $\text{Lt}_{\text{int} \times \text{int}}$, $\text{Leq}_{\text{int} \times \text{int}}$, $\text{Geq}_{\text{int} \times \text{int}}$, and $\text{Gt}_{\text{int} \times \text{int}}$ with the usual meaning of the corresponding operators $=$, \neq , $<$, \leq , \geq , and $>$ over the group of integers, respectively. For each of the predicates, either of its arguments can be an expression, attribute reference, or constant term.

4.4.13.2 Date types

For the different date types, predicates for expressing equality and order relations exist analogous to the predicates on integers, though the setting is more complex as a predicate may operate on date types of different granularities.

Relation predicates with both arguments being of the same granularized date type date^ξ with $\xi \in \mathcal{T}^{\text{date}}$ are defined for all such types. Requiring a specific type date^ξ for both arguments ensures that the comparison is *exact* because the same granularity of the arguments implies the same value domains. We also refer to this as the respective predicate having *tight comparison semantics*.

For the cases of comparing arguments with date types of different granularities, the comparison predicates are defined with more relaxed semantics considering the coarser-grained value as range and the finer-grained value as value for which the relation of the predicate has to hold with respect to the range. We refer to this as the predicate having *loose comparison semantics*. The relation predicates supporting this are defined over argument pairs with type $\text{date} \times \text{date}$. As example, consider a date of type date^{D} of day granularity and another date of type date^{h} of hour granularity. Then the predicate $\text{Eq}_{\text{date} \times \text{date}}$ on those arguments holds if and only if the latter is an hour on the day indicated by the further. This is a relaxed notion of the equality relation that matches the intuition behind the comparison of date-type arguments of different granularities. For equality, this concretely means that, if a date is expressed in a certain granularity, the intention is that for any other argument that may be finer granular and is within the range implied by the further, those are considered equal in this loose semantics. Analogously, similar ideas apply to the definition of the remaining relation predicates based on inequalities for dates as specified in the semantics in Sec. 4.11. All granularized types are direct subtypes of the `date` type—we have not defined a hierarchy over the granular types in order to ensure tight matching semantics when using the latter.

This approach of defining one generic type `date` for dates as well as multiple specific ones, date^ξ with $\xi \in \mathcal{T}^{\text{date}}$, requires us to define different semantics for the comparison relations for the different combinations of argument types. These different semantics allow parties to find their ideal tradeoff for each use case in terms of flexibility of specifying a predicate (e.g., within a policy) without imposing additional constraints on the type of a date attribute and tightness of the comparison semantics. Thus, our approach does not limit a system upfront, but leaves the choice of semantics for date operations to the parties who can decide based on the use case requirements. In a practical system, we expect that best practices will evolve and govern this aspect.

For selecting date attributes of to-be-issued identities, the above semantics needs to be kept in mind in the light of potential policies against which the identities may be presented in order to not enable system exploits based on misinterpretation of date attributes. For the same reason, policy authors must keep this generic semantics in mind when defining authorization policies.

Example 4.12 (Date granularities) *We illustrate this further with an example of a party specifying that an identity backed by a private certificate must be currently still valid in terms of its expiration date. Let the current date expressed in day granularity be 2011-07-01 and let this be used as a reference date against which the validity of the identity (certificate) c is assessed through the predicate $\text{Geq}_{\text{date} \times \text{date}}(c.\text{validUntil}, 2011-07-01)$. For an identity, the expiration date of which is encoded in day granularity with type date^D with the constant 2011-07-01, the non-expiration requirement is clearly fulfilled for a reference date of type date^D with tight (exact) comparison semantics. For the case of the identity having the hour-granular expiration date $2011-07-01-12 :: \text{date}^h$, that is, it is valid, in the strict semantics, until 12-59-59 of July 1, 2012, the inequality predicate is fulfilled as the reference date has type date , the reference value covers the whole day 2011-07-01 and loose semantics can apply. Taking this further, even after noon that day, the holder of the identity can still prove it valid in this loose semantics because the granularity of the reference date is a day. This expresses exactly the intended semantics that the identity be valid on this day and cannot capture the more fine-granular validity of the identity. This can be a problem in case there is a strong reason for the identity to be valid only until the given time that day for which reason a finer granularity has been chosen, e.g., for a short-term delegated identity valid only for some hours on a specific day. This example shows well that the granularity of an encoding might carry semantics that may be uninterpreted if used in a predicate with a certain reference value with lower granularity. Such problems can be easily mitigated by the policy author specifying a granularized date type with a specific granularity for which only comparison operations with tight semantics are possible. Our generic date semantics must also be considered in the light of ontology abstractions used in data request parts of authorization policies. Using the generic date type in a policy means that the relaxed semantics of the verification is accepted.*

We overload the predicates for equality, non-equality, and inequalities for dates to realize the different predicate signatures of interest to allow for the comparison of dates with different types and granularities and let a party choose whether an expression with fewer restrictions but looser semantics or more restrictions and tighter semantics is preferable for a specific policy rule. Note that, with our approach, we do not fix the system to one of those semantics, but leave the choice to a policy author for every single policy rule. This results in the following predicates over

arguments of the date types:

$$\left\{ \text{Eq}_{\tau_1 \times \tau_2}, \text{Neq}_{\tau_1 \times \tau_2}, \text{Lt}_{\tau_1 \times \tau_2}, \text{Leq}_{\tau_1 \times \tau_2}, \text{Geq}_{\tau_1 \times \tau_2}, \text{Gt}_{\tau_1 \times \tau_2} : \right. \\ \left. \tau_1, \tau_2 \in \left(\{\text{date}\} \cup \{\text{date}^\xi : \xi \in \{Y, M, D, h, m, s\}\} \right) \right\}$$

The language allows for each of those predicates to have as either of its arguments an expression, attribute reference, or constant. When protocols based on certificates are used for proving a formula correct, further restrictions apply on the permissible arguments due to limitations of the zero-knowledge proof protocols summarized in Chapter 6.

Note that we often do not make the types of the predicates explicit in this work, unless we emphasize which of the predicates we talk about, because mostly it is clear from the context which predicate is referred to.

Example 4.13 (Age verification) *Next, we present an example where a tight comparison semantics must be used in order to obtain the correct result in all cases, namely a concrete example use case of verification of a minimum age. The problem here is that in case a date of birth would be encoded only as the year and this value would be compared with a concrete reference date on day granularity, the loose comparison semantics may establish the required minimum age while this may not be true considering the date of birth in day granularity. The difference to expiration dates of certificates with dates of birth is that when encoding a date of birth in a lower granularity, e.g., year, then the actual date is located somewhere within this year while for an expiration date, the meaning is always that the expiration is at the end of the presented date/time unit. For this reason it is crucial that a date of birth attribute always be encoded in the same granular type with day granularity and different attributes be defined for other granularities, e.g., the attribute yearOfBirth.*

For realizing expiration date checks of identities against the current date/time or a constant offset to it, we propose a generic solution where the current date/time is represented through a system-defined *macro* `currentDateTime` in a policy (data request) that is expanded to the current date/time of one of the date types, with the granularity being determined through the predicate the macro is an argument of or the granularity of the other argument of this predicate once it becomes instantiated during the policy matching. With this approach we always achieve tight comparison semantics while not constraining a policy to a specific granularity for a date predicate. Other workable solutions are to fix the granularity of the predicate in the request to a specific granularity and through this make the policy more restrictive and possibly have fewer identity types match the policy than in the above approaches. Loose comparison semantics can be achieved trivially by using an inequality predicate over the generic type `date` and use a constant date, typed with `date`. The choice of the approach to take depends on requirements one needs to fulfill and is left to the policy author.

For validity dates of certificates, day granularity is often sufficient as it precisely mimics the granularity of many kinds of today's offline credentials, such as eID cards, passports, or public transport passes. The number of days to encode is thereby sufficiently small and leads to very efficient inequality proofs using the zero-knowledge proof protocol of Boudot [Bou00]. Doing the same with a granularity of milliseconds makes the integer encodings of the dates larger and thereby reduces the efficiency of the proofs of inequalities due to the time complexity of the algorithm used. Thus, for many of our examples in this work we revert to the day-granular date type as this is most practical for validity dates of identities as well as representation of date of birth attributes, both of which are frequently used in our example use cases as well as in practical systems.

4.4.13.3 String

For `str`-typed terms we only define the predicates $\text{Eq}_{\text{str} \times \text{str}}$ and $\text{Neq}_{\text{str} \times \text{str}}$. This is a restricted set of predicates without predicates for expressing lexicographic ordering relations, the choice of which has been governed by the requirements for privacy-enhancing identity management based on data minimization as well as what can be efficiently implemented in practice through private certificate systems for proving formulae correct.

Note that for an implementation of data release based on private certificate protocols, the $\text{Neq}_{\text{str} \times \text{str}}$ predicate on strings requires an extension of the cryptographic protocols summarized in Chapter 6 for supporting commitments of arbitrary size integers.

4.4.13.4 Boolean

For the type `bool`, only the predicates $\text{Eq}_{\text{bool} \times \text{bool}}$ and $\text{Neq}_{\text{bool} \times \text{bool}}$ are defined, with either argument of one of the predicates being an attribute reference or constant term.

4.4.13.5 Identifier

For arguments of type `idf`, the predicates $\text{Eq}_{\text{idf} \times \text{idf}}$ and $\text{Neq}_{\text{idf} \times \text{idf}}$ are defined. Either of the arguments can be an attribute reference or constant term. Because those identifiers are interpreted with themselves, equality holds exactly when the terms are equal.

4.4.13.6 Party

For `pty`-typed arguments, the `=`-relation—logical object equality—is defined, which expresses equality of the interpretations of the arguments. That is, the relation is used for expressing equality of parties referred to through party identifiers. This relation may be specified on either an attribute reference and a constant or two constants of type `pty`. Because `pty`-typed terms are not self referential, object equality can hold even if the terms are different.

Infeasibility of proving non-equality. Proving non-equality on arguments of type `pty` reliably is not possible for the reason that non-equality of parties cannot be generally proven cryptographically in our setting. For this reason, a negation of party equality cannot be proven generally and is therefore not supported to be used in a formula to be released to another party.

Concretely, an implementation of the semantics of party non-equality through credential systems based on a private key held by the party faces the problem that the party may obtain different private keys in a real system, e.g., from different registrations in the system as is possible in our open setting. Realizing non-equality of parties by equating it with the non-equality of their private keys would counter the approach of allowing for multiple registrations as in that case a party having two different keys might be the same while non-equality on the respective party terms would hold.

When making strong assumptions for a less open system than ours where parties are guaranteed to have a single master secret key and this can be reliably enforced, it may be possible to be able to prove non-equality on the holder or subject terms of identifier objects or identities while realizing proper (non-)equality semantics with respect to the underlying party. With delegation, this is hard to fully generalize because parties may have different keys in different contexts when using our approach towards delegation, where a single-key requirement for a party is hard to enforce practically in an open system.

4.4.14 Property Predicates

Another kind of predicates supported in our language are what we refer to as *property predicates*. A property predicate associates a property comprising one or more values, according to the arity of the property, with a party. There are two approaches for specifying property predicates: (1) through a single predicate, e.g., `Property` per property arity, or (2) a different property predicate for each property.

Approach (1) specifies the property by associating the property value(s) with a property name and a subject. This requires $k + 2$ different predicates for specifying any 0-ary to k -ary property predicate. The following example in (4.15) assigns the unary property value 8 of property “`trustLevel`” to a party using its subject `s`.

$$\text{Property}(s, \text{“trustLevel”}, 8) \quad (4.15)$$

Approach (2) uses a separate predicate for expressing each property, with an arity one less than the number of values for expressing the property. The example above expressed through this alternative approach is presented in (4.16).

$$\text{Trustlevel}(s, 8) \quad (4.16)$$

The premier use case for property predicates in our system is their use as part of data requests for expressing ontology-based generalizations. This requires that property predicates be specified through rule formulae in ontologies as explained in Sec. 4.10. Predicates relying on properties cannot be expressed in data statements

based on credential protocols because they cannot be directly proven to hold with those cryptographic protocols for the reason that they do not refer to any attribute—rather, property predicates are derived from attribute predicates being part of a received proven formula.

4.4.15 Other Predicates

Our system requires that another class of predicates be available as discussed next.

4.4.15.1 Registration Binding

For expressing that for an identity being referred to in a formula also holdership of its registration identity needs to be proven, we use the predicate $\text{RegBind}_{\text{id} \times \text{id}}$. Using ontology-based automated deduction as explained later, this predicate can be deduced to hold on a concrete identity and corresponding registration identity. The registration binding predicate may be used in data requests for the above-expressed purpose and can also be part of formulae derived from data requests through the \vdash -relation over an ontology.

There are multiple reasons for a data recipient to request statements over a registration identity for a specific other identity, two of which we will explain in the following. (1) The recipient does not want to rely on the certifier of the other identity that it has properly executed the check of the registration of the party in the system. Depending on whether such trust exists, obtaining proven statements about the registration identity may be required for obtaining a bullet symbol on a statement in our channel model of Chapter 3. (2) The recipient requires attributes of the registration identity to be conditionally released for reasons of accountability. Assuming attributes that are widely used in registration identities applied for the other identity are requested, the request can be phrased through the registration binding predicate. A unique object identification number, or identity serial number, or the attribute *subjectIdWithCertifier* are suitable candidate attributes for this.

In case of no such predicate being expressed in a request, it is still ensured, when using our credential protocols, that the same secret key of the party is used as was used in the issuing of the credential corresponding to the other identity. If the issuer is trusted to having done the verification of registration and binding of the other identity to the party properly, a bullet symbol can be obtained in the channel model on the proven statement without the registration identity being proven.

4.4.16 Features of an Identity

In addition to modeling user attributes and reserved attributes like the *holder* and *subject* attributes, identities may support *features* that can be executed when using the identity for releasing data.²⁵ Features are dependent on the technology underlying an identity and thus introduce a technology-dependent aspect into the modeling.

²⁵Note that in early work on system aspects of the Camenisch–Lysyanskaya credential system, the term features have a broader meaning [CH02].

In this work, we only discuss features of identities related to private certificate systems. Examples for such features are certificate revocation checking [CL02a] or limited show of certificates [CHK⁺06].

Our preferred modeling option is to express a feature as part of the data representation language. This has the advantage that the feature can be referred to in authorization policies and considered in the formal process of matching a party's identity relationships against policies, that is, the expressivity of the policy system increases towards expressing security-relevant aspects of the identity relevant for its assessment by a verifier. The inherent drawback of this approach is that policies and data statements become more complex by expressing features.

The modeling is done through reserved attributes, denoted as *feature attributes* expressing a feature's availability and properties through one or more attributes of the identity. Formulae referring to an identity can make statements about those feature attributes in identity relationships and data statements. The feature attributes are similar to any other attributes, also in terms of semantics, that is, they behave like regular attributes that can be revealed or remain hidden when making a data statement based on an identity.

Feature attributes can be specified in any part of a formula representing a data statement and the semantics of feature execution applies to the parts of the formula following standard logic semantics, which is interesting only for formulae comprising \vee -connectives and trivial for formulae only comprising \wedge -connectives.

This implies that the execution of a feature in a protocol is not part of the semantics of the data representation language as introduced in Sec. 4.11, only its specified properties are. Whether a feature is to be executed in an interaction can be expressed through *operational semantics* in such a way that whenever an attribute indicating the availability of a feature is released in a data statement, the feature is executed in the corresponding proof protocol, e.g., by executing a cryptographic protocol comprising a zero-knowledge proof protocol realizing the feature. Thus, the semantics of feature execution is modeled only operationally through appropriate implementation of the protocol and does not touch the denotational semantics of Sec. 4.11. This approach is in line with the semantics of standard logic we build upon and does not formally model the use of the feature, thereby following our paradigm of not modeling most of the technical aspects of an identity in the semantics. Also, expressing certain features using a dedicated logic-based approach would require that a non-standard logic be used for this which comes with disadvantages. Our approach completely avoids the use of non-standard logic, e.g., linear logic for expressing consumption semantics of credentials [GBB⁺06, BBG⁺07].

For an identity that is based on private certificate protocols, some practically relevant features for which cryptographic protocols exist are the *revocation*, *k-show per epoch*, and *1-show* feature.

Revocation of an identity. The possibility of an identity being revoked by its certifier is indicated through the attribute *revocation*, the value of which indicates the supported revocation method. The release of the attribute in a data statement carries the operational semantics of the feature being executed in the cryptographic

protocol used to prove the formula as follows:

$$\text{Eq}(\text{c.revocation}, \text{"dynamicAccumulators"}) .$$

Quantitative properties of the revocation feature and further details, e.g., the maximum permitted age of revocation information, can be specified through meta-data, though, not as part of the formal data model.

***k*-show per epoch of an identity.** An identity can be constrained to be shown at most k times during an *epoch*, e.g., time period, with the prover being traceable in case this number is exceeded [CHK⁺06], therefore preventing (large-scale) sharing of identities. For this feature we only express its availability through the *limitedShow* attribute as follows:

$$\text{Eq}(\text{c.limitedShow}, \text{"kShowPerEpoch"}) .$$

The predicate indicates the availability of the feature realizing the property of k -show per epoch for the identity when expressed in the corresponding object specification formula. Its release in a data statement has the operational semantics that this feature be executed through execution of a corresponding cryptographic protocol [CHK⁺06]. For an identity backed by a private certificate—the setting which this feature has been designed for—the details regarding the epoch and number of permitted uses of the identity are encoded as part of the certificate or the certificate structure.

1-show identity. Analogous to the above, 1-show identities can be specified with the *limitedShow* attribute and constant “1show” as

$$\text{Eq}(\text{c.limitedShow}, \text{"1show"}) .$$

This concludes our discussion of identity features and their implementation through operational semantics.

4.4.17 Extension: Operators

An extension to the definition of identities given earlier is that the function representing an identity maps attributes to tuples comprising a typed attribute value and *operator* from the set {eq, lt, leq, geq, gt}. The operator specifies the equality or inequality relation that holds for the attribute value, in addition to the implicit equality relation always applying in the earlier definition. The key idea of operators is that they allow for expressing inequalities directly as part of an identity without expressing a formula over its attributes, and thus adds to the expressive power. Operators can be employed also for conditionally released and opaque identities in addition to identities.

Next, we discuss some use cases for using *inequality operators* on attributes in identities, which strengthens privacy by following the data minimization principle

in a more stringent way. Generally, the certification of identities with operators other than the standard case of equality is useful in cases when the issuing of an attribute with its concrete value, that is, using the equality operator, would not reflect the targeted intention for the concerned identity and unnecessarily reveal too much information.

For example, the credit rating agency in the example below intends to issue an inequality over the salary attribute and not its precise value because this reflects what the identity is to express.

Example 4.14 (Credit rating agency) *Assume a credit rating agency assesses the creditworthiness of its customers through, among other things, their monthly salaries. Based on the assessment, it issues a customer an identity certifying its assessed creditworthiness, also comprising an attribute about the customer's salary range. For the assessment, it need not learn the customer's precise salary in order to make a statement about the rating, but it is sufficient for it to learn about a salary range in terms of a Geq predicate on the salary attribute which can be provided, for example, through a proof by the customer using one or more bank or salary statements. A credit rating identity comprising, among other attributes, the re-certified salary range that can be issued based on the proven inequality statement, or even be a different, less-revealing range, can be issued for the attribute. This approach realizes data minimization towards the certifying party in a very aggressive way by providing exactly the data required for the business process at hand and not having the party learn excessive data than that needed for the business process at hand, in a very strict sense.*

Example 4.15 (Age verification service) *Assume a service provider has specialized in issuing age proof identities that allow their holders to prove their major age without revealing their date of birth. The service provider accepts a wide range of identities or combination of identities for one's major age being proven to it as a basis for the prover getting issued the new age proof identity, that is, as breeder credentials, even such not containing the date of birth as an attribute or it being encoded differently such that it cannot be recertified through opaque identities, that is, without the service provider learning it. The service provider includes the date of birth attribute with an inequality stating that the date of birth is at least 18 years in the past, or the appropriate threshold constants for major age applying in the country at hand, from the presented issuing date of the identity using an inequality operator on the date of birth attribute instead of certifying its precise value. This approach particularly allows the party to also issue such identities when using breeder identities for the age validation that do not contain a date of birth attribute at all, though still imply an age greater than or equal to the major age threshold.*

The permissible combination of identities to establish major age is very specific, e.g., to the legislation of the country the age proof identity is to be used in and thus justifies a business case for specialized parties performing this task of identity validation, possibly taking liability for properly validating the major age of parties, and issuing new age proof identities.

Use cases like the ones above can be realized with operators expressed on attributes or, at least partially, also by re-certifying attributes of opaque identities which are not learnt by the certifying party, though which it can certify. Because of this, the examples also give a motivation for the use of opaque identities in our system. Opaque identities will only be workable for realizing the presented scenarios in case the new identity being certified is based on private certificate protocols.

Another possible—though conceptually not very clean solution in general for use cases like the above—is to implicitly express the operator as part of the attribute, e.g., integrating the inequality semantics as well as the constant into the attribute name. The problem with this approach is that using such special attribute for each inequality relation to be expressed will severely constrain certain capabilities of the resulting data representation model, e.g., matching a predicate expressed over such attribute with a data request comprising the inequality with respect to a different implicitly encoded constant. The reason for this is the mixing of different kinds of semantics, namely about the concerned attribute and a predicate expressed over it, into a single entity.

4.5 Data Statement Language

Data statements made by parties towards other parties are based on the concepts and syntax presented earlier in this chapter. Because the system we define in this thesis also gives rise to a framework, there exists not a single such language in the framework, rather, there exists one generic language $\mathcal{L}^{p/g}$ and one language per protocol for proving data statements to be released correct.

Such generic language $\mathcal{L}^{p/g}$ is based on the language that would be supported by the most expressive protocol we have in mind for our framework. Such protocol is based on online certifiers making statements about certificatees, including making statements comprising identities of multiple certifiers. This allows for better expressiveness for the language than with credential protocols as there is no need for advanced cryptographic mechanisms for ensuring integrity of such statements, though, with the weak trust model of the certifiers learning substantial information about the transactions of the certificatees. We cannot think of a class of protocols that would be more powerful in terms of expressiveness than this, thus this generic language $\mathcal{L}^{p/g}$ is the language which all statement languages for concrete underlying mechanisms are (proper) subsets of.

Any specific concrete embodiment for a data release protocol requires a (proper) fragment of $\mathcal{L}^{p/g}$ for being able to model the capabilities of the concrete mechanism. For example, the corresponding language for the abovementioned protocols with online certifiers being able to make statements about identities of multiple certifiers equals $\mathcal{L}^{p/g}$. For credential protocols, a more constrained language $\mathcal{L}^{p/c}$ needs to be defined. For conventional certificates, a substantially constrained fragment of $\mathcal{L}^{p/g}$ applies. For example, this fragment does not allow for expressing disjunctions or inequality predicates or the non-equality predicate to be expressed on attributes—only equalities for revealing attributes are supported. Thus, for each technology,

the corresponding language formalizes what the technology is capable of supporting in terms of expressiveness and data minimization.

In the remainder of this section, we present the language $\mathcal{L}^{p/c}$ for expressing data statements based on pseudonym and credential protocols as underlying technology, building on the concepts and syntactical constructs introduced earlier in this chapter. The language $\mathcal{L}^{p/c}$ is a fragment of the generic language $\mathcal{L}^{p/g}$ for expressing data statements. By design, $\mathcal{L}^{p/c}$ is constructed such that it captures the expressivity of private certificate systems in terms of privacy capabilities and integrates advanced trust management capabilities.

The expressivity of $\mathcal{L}^{p/c}$ is greater than what mainstream mechanisms can support, e.g., standard protocols with a single online certifier or traditional certificates in use as of today. Thus, the language captures, as special cases, what can be expressed with the other most prominent protocols available today, namely protocols with an online certifier and such based on traditional certificates, and combinations thereof. Protocols with online IdPs could, in certain areas, support higher expressivity, e.g., disjunctions over the type of an identity. It would be straightforward to extend the language further to accommodate this, following the overall ideas of the language design. Though, in our opinion, protocols with online certifiers are not based on a particularly privacy-friendly trust model and thus such extensions would not provide much benefit in practice.

In addition to a corresponding fragment of $\mathcal{L}^{p/g}$, each mechanism requires that various other languages be defined for specifying the input and output of the protocols required for realizing the mechanism in a system. In Chapter 5 we present the languages for credential protocols only. Further (cryptographic) mechanisms fit into our framework by design, though, are not presented in this thesis. The author discusses architectural aspects regarding the support of multiple protocols in related work [Som11].

4.5.1 Syntax

A formula ϕ of $\mathcal{L}^{p/c}$ comprises a conjunction of multiple *subformulae* as indicated in the grammar in Fig. 4.1, each of which is used for expressing certain aspects of the data statement ϕ . One subformula is used for introducing identifier objects, one for specifying third parties, one for introducing identities, one for introducing conditionally released identities, and one for expressing the actual statement over the attributes of the identifier objects, identities, and conditionally released and opaque identities. The subformula comprising the conjunction of the first four of the mentioned subformulae is referred-to as the *preamble* of the formula as it contains aspects related to the definition of the objects we are interested in referring to for making data statements. The last subformula is referred to as the *attribute assertion* or *attribute statement* subformula.

The syntax notation we employ next uses underlining for expressing productions and other elements related to the grammar notation, while non-underlined elements are ground terms of the language $\mathcal{L}^{p/c}$. We present the syntax for the subformulae of a data statement in separate subsections together with their descriptions, with

$$\phi ::= \underline{\text{IdtfObjStmts}} \wedge \underline{\text{ThirdPartySpecs}} \wedge \underline{\text{IdtyIntroduction}} \wedge \underline{\text{CRIdtyIntroduction}} \wedge \underline{\text{AttrStmts}}$$

Figure 4.1: Grammar for $\mathcal{L}^{p/c}$

common elements separated out and presented towards the end.

4.5.1.1 Identifier Object Introduction

In the first subformula of ϕ , as specified through the production $\underline{\text{IdtfObjStmts}}$ of the grammar as specified in Fig. 4.2, we express identifier assertions through which all identifier objects with the subject being the *primary subject* of the formula and, as optional addition, zero to multiple delegaters, are introduced. The primary subject is holder of all primary identifier objects, and, for the case of employing credential technologies, is also the party performing the proof and one of the parties the assertion subformula makes statements about. We refer to those identifier objects as the *primary identifier objects*.

$$\begin{aligned} \underline{\text{IdtfObjStmts}} &::= \underline{\text{IdtfObjStmt}} \wedge \underline{\text{IdtfObjStmts}} \mid \underline{\text{IdtfObjStmt}} \\ \underline{\text{IdtfObjStmt}} &::= \underline{\text{Idtf}} . \text{holder} = \underline{\text{ConsPty}} \wedge \\ &\quad \underline{\text{Idtf}} . \text{subject} = \underline{\text{ConsPty}} \wedge \text{Eq} (\underline{\text{Idtf}} . \text{subjectId} , \underline{\text{ConsIdtf}}) \\ &\quad [\wedge \text{Eq} (\underline{\text{Idtf}} . \text{domain} , \underline{\text{ConsStr}})] [\wedge \underline{\text{ObjEqPtyConsCons}}]^* \end{aligned}$$

Figure 4.2: Grammar for identifier object introduction subformulae

This subformula of ϕ serves the purpose of relating the attributes of the primary identifier objects referred to in ϕ with constant terms. For each identifier object, its *holder* and *subject* must be associated with a constant of type *pty*, its attribute *subjectId* with a constant of type *idf*, and, in case of a domain identifier object, the *domain* attribute with a *str*-typed constant. For each identifier object, the *subjectId* is the pseudonym value, while the *holder* and *subject* attributes are used for introducing the associated terms into the formula, binding the identifier object to its holder and subject, and, in later parts of ϕ , for binding further objects to their holders and subjects.

The optional production $\underline{\text{ObjEqPtyConsCons}}$ specifies the logical equality relations between *pty*-typed terms for stating holder or subject equality for different identifier objects in the same registration domain or bridging registration domains. That is, a constant t_1 of type *pty* associated with the holder or subject attribute of an identifier object as explained can be related through object equality with another such constant t_2 for relating *pty*-terms representing the holder or subject of

different identifier objects through logical equality, particularly also objects from different registration domains—the details of the use of such object equality in this context, particularly also the use of different registration domains, are discussed in Sec. 4.8.2. Such relation of terms results in the logical equality relation $t_1 = t_2$ between the terms.

Note that relaxing the ordering of predicates in the preamble subformulae is inline with the ideas behind the language. For better readability, though, we partition the overall formula into its subformulae and constrain the ordering of predicates.

The language does not permit one to express predicates over attributes of primary identifier objects in any other parts of the formula, particularly not in disjunction branches. The reason for the latter is that this would not allow for useful functionality either in terms of identity management or privacy. For each identifier object a proof is made over, the data recipient knows the corresponding shared cryptographic information, thus expressing a disjunction over the attributes cannot help data minimization.

4.5.1.2 Third-party Specification

Third parties can be specified through statements over attributes of identifier objects and identities which they are subjects of. Such identifier objects and identities are referred to as *secondary identifier objects* and *secondary identities*, respectively. Currently required classes of third parties are certifiers of identities and conditional data recipients associated with conditionally released identities. A third party of either kind is specified through either of the approaches as we explain next. The syntax for the third-party specifications of a formula is specified through the production *ThirdPartySpecs* of the grammar presented in Fig. 4.3.

Most trivially, one can omit a third-party specification statement for a third party being a certifier or conditional data recipient which is always referred to in other parts of the formula through a subject term, in which case the third party is specified only through this subject term.

Another means of specifying the third party further is to make a statement about it being the subject of an identifier object which is then further specified through its *subjectId*. The holder can, though, need not, be expressed for third-party identifier objects, because the holder is, in the protocols we consider, always the same third party already specified through the subject.

Yet another means is to specify the third party through statements over the attributes of an identity with the third party as subject. Like for the case of identifier objects, the holder is again the third party and thus can be omitted from the specification.

The approach of using an identifier object and identity can also be combined for the specification of the third party. Also, multiple identifier objects and identities can be introduced and related to each other for specifying the third party. This may require that the subject terms associated with them be related through object equality relations. Multiple objects are useful for exploiting the information of multiple public-key or other certificates about the third party in a single statement.

The identity with the third party as the subject has, much like any other identity, a certifier associated. This can be used for specifying this certifier as being another third party, using the exactly same concepts recursively. Such recursion can be terminated by declaring a party as a trusted Certification Authority (CA) or not making a further statement about it. The very general way of stating a certification chain explicitly may be relevant in special cases only, the regular case is to specify the third party through an identifier object or identity or combination thereof. Practically, referring to a unique public pseudonym is often sufficient—all other information about the third party is not strictly required as it is known by the recipient of the formula due to the object specification formulae of objects related to third parties the recipient holds for verifying proofs. Particularly the approach of using multiple objects and recursive specification of the certifier for specifying a third party is mainly interesting in the data request language $\mathcal{L}^{r/c}$ dual to $\mathcal{L}^{p/c}$.

When using multiple identifier objects or identities with the same third party as subject, terms referring to the party are handled as discussed in Sec. 4.8.

4.5.1.3 Identity Introduction

The identity introduction subformula of a formula ϕ handles the introduction of primary identities, with a syntax as specified through the production IdtyIntroduction as presented in Fig. 4.4. That is, each identity with the primary subject or a delegator as subject that is to be later referenced in the formula needs to be introduced at this point through specifying crucial aspects related to it. This comprises specification of the certifier term and the type of the identity. For credential protocols, both those must be specified in the top-level \wedge -node of a data statement formula, that is, no disjunctions are allowed in this subformula, as the specification must uniquely determine the third party and thus the public key to be used for verifying the parts of the underlying cryptographic protocol related to the corresponding private certificate.

Note that the association of the holder and subject of an identity with terms referring to parties is done only in the attribute statement subformula for the reason of being able to realize disjunction statements within the latter. Fixing those associations already in the top-level \wedge -node of ϕ , e.g., as part of the identity introduction subformula, would prevent us from expressing disjunctions over predicates in the attribute statement subformula in a general way. Note also that registration domain bridging identities, that is, identities for the purpose of bridging two registration domains, must be introduced in the identity introduction subformula in case their domain bridging capability is used through an according logical equality relation over constant `pty`-typed terms related to identifier objects of different registration domains given in the identifier object introduction subformula.

4.5.1.4 Conditionally Released Identity Introduction

This subformula introduces all conditionally released identities referred to in the attribute statement subformula of ϕ . The purpose is to fix the conditional recipient

$$\begin{aligned}
\text{ThirdPartySpecs} &::= \text{ThirdPartySpec} \wedge \text{ThirdPartySpecs} \mid \text{ThirdPartySpec} \\
\text{ThirdPartySpec} &::= \text{IdtfObjStmts3P} \wedge \text{IdtyStmts3P} \wedge \text{IdtfObjEqPtyAttrCons}^* \\
\text{IdtfObjStmts3P} &::= \text{IdtfObjStmnt3P} \wedge \text{IdtfObjStmts3P} \mid \text{IdtfObjStmnt3P} \\
\text{IdtfObjStmnt3P} &::= \text{Idtf} . \text{holder} = \text{ConsPty} \mid \text{Idtf} . \text{subject} = \text{ConsPty} \mid \\
&\quad \text{Eq} (\text{Idtf} . \text{subjectId} , \text{ConsIdtf}) \mid \\
&\quad \text{Eq} (\text{Idtf} . \text{domain} , \text{ConsStr}) \mid \text{ObjEqPtyConsCons} \\
\text{IdtyStmts3P} &::= \text{IdtyStmnt3P} \wedge \text{IdtyStmts3P} \mid \text{IdtyStmnt3P} \\
\text{IdtyStmnt3P} &::= \text{AttrPred3P} \wedge \text{IdtyStmnt3P} \mid \text{AttrPred3P} \\
\text{AttrPred3P} &::= \text{AttrRelPred3P} \mid \text{IdtyEqPtyAttrCons} \mid \\
&\quad \text{Property} (\text{ConsPty} , \text{"trustedCA"}) \\
\text{AttrRelPred3P} &::= \text{PredInt3P} \mid \text{PredDate3P} \mid \text{PredStr3P} \mid \text{PredBool3P} \mid \text{PredIdtf3P} \\
\text{PredInt3P} &::= \text{PredSymInt} (\text{ExprInt3P} , \text{ExprInt3P}) \\
\text{ExprInt3P} &::= \text{ConsInt} \cdot \text{AttrRef3P} \mid \text{AddOpInt} \mid \text{ExprInt3P} \mid \text{ConsInt} \\
\text{PredDate3P} &::= \text{PredSymDate} (\text{ExprDate3P} , \text{ExprDate3P}) \\
\text{ExprDate3P} &::= \text{ConsDate} \mid \text{AttrRef3P} \mid \text{AttrRef3P} \mid \text{AddOpDate} \mid \text{ConsInt} \mid \\
&\quad \text{ConsDate} \mid \text{AddOpDate} \mid \text{ConsInt} \\
\text{PredStr3P} &::= \text{PredSymStr} (\text{ExprStr} , \text{ExprStr}) \\
\text{ExprStr} &::= \text{AttrRef3P} \mid \text{ConsStr} \\
\text{PredBool3P} &::= \text{PredSymBool} (\text{ExprBool} , \text{ExprBool}) \\
\text{ExprBool} &::= \text{AttrRef3P} \mid \text{ConsBool} \\
\text{AttrRef3P} &::= \text{Obj3P} . \text{Attr} \\
\text{Obj3P} &::= \text{Idty}
\end{aligned}$$

Figure 4.3: Grammar for third party specification subformula

$$\text{IdtyIntroduction} ::= \text{Idty} . \text{certifier} = \text{ConsPty} \wedge \text{Eq} (\text{Idty} . \text{type} , \text{ConsStr})$$

Figure 4.4: Grammar for identity introduction subformula

as well as the condition associated with a conditionally released identity throughout the formula. This is required, analogous to fixing the certifier and type of identities, because the cryptographic protocol for realizing conditionally released identities [CS03] requires that the used public key for encryption and the condition be fixed throughout the formula. The syntax of the subformula is specified through the CRIdtyIntroduction production presented in Fig. 4.5.

$$\begin{aligned} \underline{\text{CRIdtyIntroduction}} ::= & \underline{\text{CRIdty}} . \textit{recipient} = \underline{\text{ConsPty}} \wedge \\ & \text{Eq} (\underline{\text{CRIdty}} . \textit{condition} , \underline{\text{ConsStr}}) \end{aligned}$$

Figure 4.5: Grammar for conditionally released identity introduction subformula

The constant specifying the recipient can be related to the subject of one or more identifier objects or identities in the third-party specification subformula, over which further predicates can be expressed as explained. Except for the attribute *recipient*, other attributes of a conditionally released identity must never be related to *pty*-typed attributes or constants in any place in the formula. Note that the requirement of fixing the condition for the complete formula can be relaxed when using the Camenisch–Damgård [CD00] verifiable encryption scheme instead of that of Camenisch and Shoup [CS03] we support as only scheme in our system.

Note also that opaque identities need not be introduced in the preamble of the formula as their cryptographic key material is derived in the protocol and thus need not be specified explicitly—it is sufficient to associate their attributes to that of other objects in the attribute specification subformula.

4.5.1.5 Attribute Assertions

The attribute assertions subformula specifies the attribute assertions, or attribute statements, to be expressed over the introduced identifier objects, identities, conditionally released identities, and zero or more opaque identities. The subformula thereby relates attributes of those objects with each other or constants through the relation predicates for the respective types. Predicates can be arranged in an arbitrary structure connected through the logical \wedge and \vee operators. Certification metadata other than the certifier specification, e.g., expiration of identities, are expressed using the same syntax as for regular attributes. The syntax for the attribute assertions subformula is given through the production AttrStmts presented in Fig. 4.6.

Logical equality relations can be used for associating the *holder* and *subject* attributes of identities with constant terms introduced in the identifier object introduction subformula. This realizes the binding of primary identities to primary identifier objects and thus parties the formula is making statements about. Those logical equality predicates should, by convention, appear in the topmost \wedge -node of the attribute assertions subformula where this is possible for a given identity in

order to not require repetition of this in different disjunctive branches.²⁶ Note that the bridging of different registration domains or delegation registration domains is accounted for in the identifier object introduction subformula.

$$\begin{aligned}
\text{AttrStmts} &::= \text{AttrPred} \mid \text{LogicalConn} \mid \text{AttrStmts} \mid \text{AttrPred} \\
\text{AttrPred} &::= \text{AttrRelPred} \mid \text{IdtfEqPtyAttrCons} \\
\text{AttrRelPred} &::= \text{PredInt} \mid \text{PredDate} \mid \text{PredStr} \mid \text{PredBool} \mid \text{PredIdtf} \\
\text{LogicalConn} &::= \wedge \mid \vee
\end{aligned}$$

Figure 4.6: Grammar for attribute assertions subformula

In Fig. 4.7 we present the integer and date predicates including the expression syntax of the arguments.

$$\begin{aligned}
\text{PredInt} &::= \text{PredSymInt} (\text{ExprInt} , \text{ExprInt}) \\
\text{PredSymInt} &::= \text{Eq}_{\text{int} \times \text{int}} \mid \text{Neq}_{\text{int} \times \text{int}} \mid \text{Lt}_{\text{int} \times \text{int}} \mid \text{Leq}_{\text{int} \times \text{int}} \mid \text{Geq}_{\text{int} \times \text{int}} \mid \text{Gt}_{\text{int} \times \text{int}} \\
\text{ExprInt} &::= \text{ConsInt} \mid \text{MulOpInt} \mid \text{AttrRefAttrAssrt} \mid \text{AddOpInt} \mid \text{ExprInt} \mid \text{ConsInt} \\
\text{PredDate} &::= \text{PredSymDate} (\text{ExprDate} , \text{ExprDate}) \\
\text{PredSymDate} &::= \text{Eq}_{\text{date} \times \text{date}} \mid \text{Neq}_{\text{date} \times \text{date}} \mid \text{Lt}_{\text{date} \times \text{date}} \mid \\
&\quad \text{Leq}_{\text{date} \times \text{date}} \mid \text{Geq}_{\text{date} \times \text{date}} \mid \text{Gt}_{\text{date} \times \text{date}} \\
\text{ExprDate} &::= \text{ConsDate} \mid \text{AttrRefAttrAssrt} \mid \text{AttrRefAttrAssrt} \mid \text{AddOpDate} \\
&\quad \text{ConsInt} \mid \text{ConsDate} \mid \text{AddOpDate} \mid \text{ConsInt}
\end{aligned}$$

Figure 4.7: Grammar for integer and date predicates

The syntax for predicates for the remaining data types `str`, `bool`, and `idf` is presented in Fig. 4.8.

Syntactical elements referred in productions earlier and that have not been introduced yet are presented in Fig. 4.9. Note that the multiplicative and additive operators are, for brevity, often referred to without their type annotation ^{int} or ^{date}.

4.5.1.6 Additional Syntax Elements

We have focused on the most important aspects of the language in the presentation of the syntax, while leaving out certain elements, the precise specification of which is

²⁶This cannot be handled in the top-level \wedge -node of the formula for the reason that for a formula with disjunctions, the non-fulfilled parts of the formula could not be cryptographically simulated in all cases due to the association of the holder or subject.

$$\begin{aligned}
\underline{\text{PredStr}} &::= \underline{\text{PredSymStr}} (\underline{\text{ExprStr}} , \underline{\text{ExprStr}}) \\
\underline{\text{PredSymStr}} &::= \underline{\text{Eq}}_{\text{str} \times \text{str}} \mid \underline{\text{Neq}}_{\text{str} \times \text{str}} \\
\underline{\text{ExprStr}} &::= \underline{\text{AttrRefAttrAssrt}} \mid \underline{\text{ConsStr}} \\
\underline{\text{PredBool}} &::= \underline{\text{PredSymBool}} (\underline{\text{ArgBool}} , \underline{\text{ExprBool}}) \\
\underline{\text{PredSymBool}} &::= \underline{\text{Eq}}_{\text{bool} \times \text{bool}} \mid \underline{\text{Neq}}_{\text{bool} \times \text{bool}} \\
\underline{\text{ExprBool}} &::= \underline{\text{AttrRefAttrAssrt}} \mid \underline{\text{ConsBool}} \\
\underline{\text{PredIdtf}} &::= \underline{\text{PredSymIden}} (\underline{\text{ArgIden}} , \underline{\text{ExprIdtf}}) \\
\underline{\text{PredSymIden}} &::= \underline{\text{Eq}}_{\text{idf} \times \text{idf}} \mid \underline{\text{Neq}}_{\text{idf} \times \text{idf}} \\
\underline{\text{ExprIdtf}} &::= \underline{\text{AttrRefAttrAssrt}} \mid \underline{\text{ConsIdtf}}
\end{aligned}$$

Figure 4.8: Grammar for string, boolean, and identifier predicates

$$\begin{aligned}
\underline{\text{MulOpInt}} &::= \cdot^{\text{int}} \\
\underline{\text{AddOpInt}} &::= +^{\text{int}} \mid -^{\text{int}} \\
\underline{\text{AttrRefAttrAssrt}} &::= \underline{\text{ObjAttrAssrt}} . \underline{\text{Attr}} \\
\underline{\text{ObjAttrAssrt}} &::= \underline{\text{Idty}} \mid \underline{\text{CRIIdty}} \mid \underline{\text{Oldty}} \\
\underline{\text{ObjEqPtyConsCons}} &::= \underline{\text{ConsPty}} = \underline{\text{ConsPty}} \\
\underline{\text{IdtfEqPtyAttrCons}} &::= \underline{\text{IdtfAttrRefPty}} = \underline{\text{ConsPty}} \mid \underline{\text{ConsPty}} = \underline{\text{IdtfAttrRefPty}} \\
\underline{\text{IdtfAttrRefPty}} &::= \underline{\text{Idtf}} . \underline{\text{IdtfAttrPty}} \\
\underline{\text{IdtfAttrPty}} &::= \text{holder} \mid \text{subject} \\
\underline{\text{IdtyEqPtyAttrCons}} &::= \underline{\text{IdtyAttrRefPty}} = \underline{\text{ConsPty}} \mid \underline{\text{ConsPty}} = \underline{\text{IdtyAttrRefPty}} \\
\underline{\text{IdtyAttrRefPty}} &::= \underline{\text{Idty}} . \underline{\text{IdtyAttrPty}} \\
\underline{\text{IdtyAttrPty}} &::= \text{holder} \mid \text{subject} \mid \text{certifier}
\end{aligned}$$

Figure 4.9: Grammar for additional syntactical elements

of less interest here. This comprises the details of the composition of ground terms of the language, such as the definitions of the productions corresponding to `pty`, the basic data types `int`, `date` and the variants of the latter, `str`, `bool`, and `idf`, or the definition of the permissible ground terms through the productions `ldtf`, `ldty`, `CRldty`, and `Oldty` referring to identifier objects, identities, conditionally released identities, and opaque identities, respectively. That is, the corresponding productions referred above are not further defined. Particularly, the different date types have been abstracted in the syntax as they are all handled analogous as the type `date` for which the productions have been presented.

The `< >` annotation for predicates has not been captured in the formal syntax. This special language element has been introduced for supporting the cryptographic proof of statements comprising disjunctions. Every predicate that is annotated with the `< >` annotation, is required to be backed with cryptographic material, e.g., private certificates. Other predicates need not hold due to being comprised in disjunction branches such that the formula is satisfied by only the `< >`-annotated predicates being true. The purpose of the `< >` annotation of a formula is to instruct the prover-side cryptographic protocol on which parts of a formula to prove with available cryptographic material and which to simulate. The annotation is removed after computing (the parts of) the cryptographic protocol for generating a proof of the formula and before the formula is revealed to the data recipient. This annotation approach has been introduced in earlier work by the author [CSZ06b, CSZ06a].

Parentheses (and) for defining precedences in the formula have not been included in the syntax either and are defined as is done commonly for related first-order logic languages.

4.5.2 Interoperability Aspects

An important aspect of an implementation is how the names of attributes are realized regarding the used namespace. It is crucial that different parties do not define the same attribute name for different purposes. A properly chosen namespace resolves this issue and can allow the parties in the system to select attribute names only from namespaces they control. The URI scheme [BLFM05] of the W3C is a practical and scalable namespace scheme that is suitable for our purposes. Analogous considerations apply for types of identities that should thus be chosen accordingly.

In the context of earlier work in this area, experiments have been performed for modeling and implementing a previous—and less powerful—variant of our data representation language based on W3C's RDF [ME04] and OWL [MvHE04] languages because of the existing tools for those. In the current presentation of the architecture we do not constrain ourselves to any specific technology by using generic first-order logic with the option of realizing a constrained system based on the presented approach.

4.6 Data Request Language

An authentication system following our model of Chapter 3 requires that authorization policies be expressed and communicated by a party to communication partners. Authorization policies express, among other things, requirements that an access requester needs to fulfill in order to gain access to a protected resource. Those data requirements are expressed as one key part of such policies. In this section, we discuss a language $\mathcal{L}^{r/c}$ for specifying data requests based on our data model. We will use the terms data request, authorization policy, or policy interchangeably when the meaning is clear from the context.

The language $\mathcal{L}^{r/c}$ is dual to $\mathcal{L}^{p/c}$ in that formulae of $\mathcal{L}^{p/c}$ can be valid responses to a formula in $\mathcal{L}^{r/c}$. As for $\mathcal{L}^{p/c}$, a generic language $\mathcal{L}^{r/g}$ is the most generic request language dual to $\mathcal{L}^{p/g}$. For every mechanism for releasing certified data that is to be supported in our framework, a pair comprising a request language and dual statement language needs to be specified and considered for the process of matching a party's object specification formulae in its repository with policies.

A data statement expressed in $\mathcal{L}^{p/c}$ is characterized by being a concrete formula without free variables, referring to concrete identities and other objects which it makes statements about. A data request is a formula of $\mathcal{L}^{r/c}$ which has the meaning that it can be fulfilled by any one of a potentially large set of data statement formulae, and, it does not make a statement about objects, rather it has the meaning of a request of statements. Requests are syntactically expressed in a related way to data statements, though with important differences that lend them their data request characteristics.

4.6.1 Syntax

The data request syntax is based on the syntax of the data formula language—we next explain the differences of the request to the statement language, though, do not present the complete syntax in formal notation to avoid repeating aspects already presented earlier. Particularly, a request formula comprises sub-formulae analogous to that of a statement formula.

A main concept of the request language is to use *free variables* instead of constant terms, e.g., for referring to objects or attribute values. A free variable stands, following standard logic, for any of a (potentially large) set of terms that the variable can be instantiated with in a statement fulfilling the request. A free variable thus realizes the general request semantics of referring to a set of matching instances instead of only a concrete single instance.

Free variables are used for referring to different kinds of objects to realize the request semantics of the formula. For example, the primary identifier objects and identities of a formula are referred to with free variables instead of constants as in a statement, meaning that any concrete identifier objects or identities fulfilling the specified constraints can be used in a corresponding data statement. Similarly, conditionally released and opaque identities can be represented through free variables. When not referring to a concrete third party and abstractly specifying a set

of applicable third parties, either for being a certifier or conditional recipient, the third-party identifier objects or identities can be referred to through free variables.

Requests of attribute values are expressed by specifying a predicate stating equality through the **Eq** predicate of the respective type over the attribute of an object (variable) as one argument and a free variable instead of a constant representing the attribute value as the other argument.

Parties can be referred to through free variables of type **pty** instead of through constants. For associating a **pty**-typed attribute with a free variable, a logical equality relation is expressed between the attribute reference and a free variable.

A request can use property predicates, with their underlying ontology-based abstractions, for associating properties with parties, where the parties are represented through free variables. Property predicates can be applied for abstracting the specification of third parties or any other party referred to in the request through abstract, ontology-defined, properties. Property predicates are expressed with the following syntax, where TermPty refers to either a constant or free variable, where the latter is useful for expressing generalizations, and ConsBasicType to a constant of a basic data type:

$$\text{PropertyPred} ::= \text{Property} (\text{TermPty} , \text{ConsStr} [, \text{ConsBasicType}]^*) .$$

A trivial, degenerate, form of data requests exactly uses the syntax of data statement formulae and thus does not comprise free variables. Such requests resemble a set of size 1 of formulae that can fulfill it—the request itself is its only fulfilling statement. Despite being formally valid, those requests are a special case and not very interesting in terms of the language and for practical applications as such requests are tailored to specific identifier objects or identities and thus to a specific requester and not generally applicable for requesting authentication from any potential service accessor.

Identifier object introduction. The identifier object introduction sub-formula is expressed analogous to the corresponding sub-formula in a data statement, with the difference that variables can be—and are typically—used instead of only constants for referring to the identifier objects and the values of their attributes. Through this, the general request semantics of referring to a set of eligible objects rather than only specific objects is realized. An identifier object referred to through a free variable and its attributes being equated to free variables refers to the set of the identifier objects other parties have established with the originator of such request.

Regarding different registration domains, a request can be phrased to refer to a party with a single **pty**-typed term, or multiple such, related through equality relations, reflecting multiple possible registration domains. Both approaches can result in logically equivalent requests and can thus be fulfilled through the same responses. If not every potential registration domain is reflected through a different identifier for a party, a responding data statement may require the introduction of identifier objects with no corresponding object in the request into the response, which is valid

as it corresponds to the release of additional information to what is requested. As the partition of identifier objects and identities of a potential respondent to a request is not known in general by the requester, a request should not make the registration domains explicit and allow respondents to respond with a formula with objects from multiple bridged registration domains, for not unnecessarily constraining the set of eligible responses.

Third-party specification. The third-party specification sub-formula in a request builds on multiple generalizations—in addition to using free variables instead of constant terms—over the third-party specification sub-formula of data statements.

Predicates expressed over the identifier objects and identities of the third party through their attributes can be connected with disjunctions instead of only conjunction. Furthermore, property predicates can be used for specifying properties of third parties through ontology concepts. A property predicate is expressed over the subject variable referring to the third party to be specified. Both approaches can be combined for further generalizing the expressiveness for requests. A property predicate over a `pty`-typed variable can represent a possibly large set of parties, disjunction over multiple such resulting in the union of the corresponding sets.

A simple third-party specification sub-formula of a request specifying a concrete, well-defined third party can do so through the public pseudonym of the third party of one of its identifier objects or identifying attributes of an identity the third party is subject of. The simplest way of specifying a third party is not having a third-party specification for this party, though, only specifying it through a unique `pty`-typed term related to the certifier or conditional recipient attribute of an object or conditionally released identity.

Using the approach of recursively specifying the certifiers of third-party identities may be interesting for specifying third parties based on their properties or that of the certifiers of identities comprising attributes those properties can be deduced from.

Identity introduction. Like in a data statement, identities are introduced through a sub-formula. Though, a request can be substantially more generic than a statement in that it does not need to uniquely specify an identity type, certifier, or even concrete identity. Rather, it can specify a requested identity abstractly such that identities of types with certain criteria certified by certifiers with certain criteria, as specified in the third-party specification sub-formula, can match it.

The type of an identity can be specified through a type defined in a subtype ontology in place of a concrete type as used in a statement. Using further ontology concepts can be done easily in our language without any conceptual extensions other than requiring an ontology making statements about such concepts. This is an advantage related to building on extensions of first-order logic and calculi. Not constraining the type of the identity in any way leaves the type open to dynamic typing as introduced in Sec. 4.4.11.2 for finding matching identities.

Disjunctions over logical equality predicates over the certifier attribute with different subject variables for the certifier can be expressed. The same semantics of

expressing a disjunction over certifiers can be achieved through specifying a disjunction in the third-party specification sub-formula over a single subject variable for the certifier. Both approaches can be combined in practical data requests for increased flexibility.

By convention, the static type of an identity should be specified through the **Type** predicate instead of the *type* attribute of the identity in order to leverage the subtyping scheme for identities of Sec. 4.4.11.2. Type and certifier predicates can be mixed in a sub-formula for an identity in order to have very generic expressivity of also being able to express dependencies between type and certifier. Disjunctions can be expressed over the predicates in the sub-formula.

A predicate **RegBind** is defined for requesting the corresponding registration binding identity C_R of an identity C as presented in (4.17). Expressing registration binding semantics without such predicate would require a disjunction over all possible kinds of registration identities for C . Such predicate is always comprised in the top-level \wedge -node of the formula. The usual concepts can be applied for further specifying C_R . Further specification of the registration identity is not required in the standard case of requesting it, unless specific constraints need to be imposed on it.

$$\text{RegBind}(C, C_R) \tag{4.17}$$

Conditionally released identity introduction. Next, we assume that **B** has specified a request formula and **A** is a potential respondent. Opaque and conditionally released identities **A** has established with **B** in earlier instances of a protocol for releasing data as specified through data statements can be referenced through the constant terms shared between **A** and **B** referring to those objects. Free variables of type *crid* or *oid* are used either for indicating that such objects be established in the protocol corresponding to a fulfilling response, or that any previously generated such object matching the request be used. Which of the two should apply in a request cannot be expressed in our logic and needs to be communicated as metadata associated with the constant or variable term referring to the (to-be-generated) object. Note, though, that the meaning of a variable is the same as always—it can be instantiated with any suitable object in a response, with the difference that potentially this object does not exist yet.

Attribute statement. The attribute assertion sub-formula is specified analogous to a data statement, with the possibility of using free variables for referring to objects and attributes using the standard request semantics. This sub-formula comprises the actual request for attribute data about a user and optionally delegaters, as well as the specification of attribute values of uninstantiated conditionally released and opaque identities.

4.6.2 Meaning of Disjunctions and Ontological Concepts

The formal-logic-based language for expressing requests has the following inherent limitation: A request which specifies an identity through its type or its certifier through abstract ontology-based concepts such that a set of concrete identity types of possibly different certifiers fulfills this specification and requests proof of \wedge -connected and \vee -connected predicates in the attribute statement sub-formula over such identity can—from a perspective of the underlying logic—be fulfilled in multiple ways: (1) Through the predicates referring to a single such identity type of a concrete certifier in the attribute statement sub-formula, or (2) through transforming the part of the attribute sub-formula comprising the aforementioned predicates such that disjunction semantics with respect to (a subset of) the permissible identities of different types and certifiers is expressed over this part of the attribute sub-formula. This applies analogously for the similar case of the request comprising a disjunction over sub-formulae explicitly instead of the ontology concept: Such request can equally be fulfilled by fulfilling the disjunction through a satisfying assignment or by proving the disjunction correct as a whole.

A possible means of expressing requirements on whether it shall be permissible that such or similar requests be fulfilled with a disjunction or whether they must be fulfilled otherwise is to use metadata outside of the logic-based language or extend the language—the language itself cannot express those requirements. In our view, such semantics technically is outside of first-order logic and thus should be modeled through metadata.

We want to point out that when specifying an identity in a request in a generic manner through using ontology concepts and disjunctions, not every attribute referred to through qualification of the identity variable needs to be contained in every identity in the set of eligible identities. Dynamic typing rules out the non-applicable identities when finding a fulfilling formula for the request. Otherwise, this would be a severe limitation of our approach for practical systems.

Properties. In the general case, there is, also due to using ontology-based abstractions, no bijective correspondence between free variables in a request and constant terms in a response. For example, a single property predicate may require a subformula with a conjunction of multiple predicates over attributes for fulfilling it. As another example, a disjunction in a request over multiple predicates expressed over attributes of different free variables representing identities may be fulfilled with a single predicate over an attribute of one concrete identity following the semantics of first-order logic.

A data request ψ represents the set Φ^{rsp} of formulae which are capable of fulfilling the request. For a data statement $\phi \in \Phi^{\text{rsp}}$ fulfilling a data request ψ we require that it contain no free variables. An *environment* is, following standard terminology in logic [HR04], a mapping from all free variables of a (request) formula to constant terms.

A statement $\phi \in \Phi^{\text{rsp}}$ *fulfills* a request ψ , with \mathcal{O} an ontology and \mathcal{E} an environ-

ment instantiating free variables of ψ , if and only if the following holds:

$$\phi \vdash_{\mathcal{O}}^{\mathcal{E}} \psi . \quad (4.18)$$

Definition 4.2 (Equivalence of requests) *Two requests ψ_1 and ψ_2 are equivalent with respect to an ontology \mathcal{O} in a deductive system \mathfrak{L} if and only if they have the same sets Φ_1 and Φ_2 of data statement formulae and corresponding environments, where, for $i \in \{1, 2\}$, each formula $\phi_{k,i} \in \Phi_i^{\text{sp}}$ with its corresponding environment $\mathcal{E}_{k,i}$ fulfills the respective request, that is, for which the sequent $\phi_{k,i} \vdash_{\mathcal{O}}^{\mathcal{E}_{k,i}} \psi_i$ is fulfilled.*

4.6.3 Examples

We next give some simple examples for data requests as well as suitable responses to the requests.

Example 4.16 (Data request) *The following example is a fragment of a request ψ requesting the `firstName` attribute as well as a proof that a given predicate holds over the monthly salary of the party and that the account is in currency Euro, all based on the same identity C of type “bank_Statement”. Note the free variable C standing for the identity as well as `FirstName` for the value of the requested attribute. The certifier is specified through its subject constant “c4112” for a concrete party.*

$$\begin{aligned} & \dots \text{Eq}(C.\text{type}, \text{“bank_Statement”}) \wedge C.\text{certifier} = \text{“c4112”} \dots \\ & \dots \text{Eq}(C.\text{firstName}, \text{FirstName}) \wedge \text{Geq}(C.\text{monthlySalary}, 3500) \wedge \\ & \quad \text{Eq}(C.\text{currency}, \text{“EUR”}) \wedge \dots \end{aligned}$$

A response to such a request needs to follow the rule that each of the request’s free variables needs to be instantiated in the response formula. The free variables are C representing an identity as well as `FirstName` representing the attribute value of the `firstName` attribute of C .

Example 4.17 (Minimal data response) *The following statement ϕ_1 is a fragment of a proper response to ψ based on the identity `bs`:*

$$\begin{aligned} & \dots \text{Eq}(\text{bs.type}, \text{“bank_Statement”}) \wedge \text{bs.certifier} = \text{“c4112”} \dots \\ & \dots \text{Eq}(\text{bs.firstName}, \text{“Jane”}) \wedge \text{Geq}(\text{bs.monthlySalary}, 3500) \wedge \\ & \quad \text{Eq}(\text{bs.currency}, \text{“EUR”}) \dots \end{aligned}$$

Example 4.18 (Data response) *So is the following statement fragment ϕ_2 , exposing more information about the salary, but still fulfilling ψ and thus being a valid response to it:*

$$\begin{aligned} & \dots \text{Eq}(\text{bs.type}, \text{“bank_Statement”}) \wedge \text{bs.certifier} = \text{“c4112”} \dots \\ & \dots \text{Eq}(\text{bs.firstName}, \text{“Jane”}) \wedge \text{Eq}(\text{bs.lastName}, \text{“Doe”}) \wedge \\ & \quad \text{Geq}(\text{bs.monthlySalary}, 9200) \wedge \text{Eq}(\text{bs.currency}, \text{“EUR”}) \dots \end{aligned}$$

The response ϕ_2 releases, in addition to what is requested, the value of a further attribute of the identity and more information on the monthly income.

Example 4.19 (Request for proving identifier object holdership) *A fragment for a request for proving holdership of an identifier relationship can be formalized as follows in our language, making use of the identifier variable P and relating its attributes to free variables:*

$$\dots P.\text{holder} = \text{Holder} \wedge P.\text{subject} = \text{Subject} \wedge \text{Eq}(P.\text{subjectId}, \text{SubjectId}) \dots$$

Note that variable names in requests should be chosen according to a scheme to avoid accidental clashes of them in data request parts of authorization policies the data requests of which may be combined to a request to be sent to the other party.

Example 4.20 (Holdership of an electronic Swiss passport) *The following example is a request for holdership of an electronic Swiss passport without revealing any of its data attributes. The system-defined macro `currentDateTime` gets expanded to the current date within the protocol for releasing the formula to the other party.*

$$\begin{aligned} &\dots \text{Eq}(P.\text{type}, \text{"swissPassport"}) \wedge P.\text{certifier} = \text{"c4112"} \dots \\ &\dots \text{Geq}(P.\text{validUntil}, \text{currentDateTime}) \dots \end{aligned}$$

Requesting or expressing the fact of holding an identity of a specific identity type without making any statements over its user attributes is a frequently required operation in our view as many permissions in real-world scenarios can be expressed through holdership of an identity of a certain type certified by a specific party. This can be expressed by specifying the type of the identity and optionally possibly further metadata contained therein such as the temporal validity of certification. Expressing holdership of an identity usually requires that at least the type of the identity be given as otherwise any identity would be suitable and thus insufficient information about the identity would be available for an authorization decision. This integrates well with our model without the need of introducing a new language concept, because we have the type and other metadata expressed as attributes of identities, equal to user attributes.

Example 4.21 (Entrance ticket) *Another example where holdership and basic attributes of an identity can be sufficient is for an entrance ticket for entering a dance club: Only the type, certifier, and temporal validity of the ticket may be sufficient for a decision on granting or denying the holder of the identity entrance to the club.*

When thinking of a widespread use of an electronic identity system as ours, many more examples of similar cases of authorization policies can be thought of. Note that proving holdership can also be done with only revealing the certifier while hiding the type, though this is possible for certain protocols only for which the type is, unlike for private certificate protocols, not required for producing a proof for the statement.

4.6.4 Third-party Requests

The default data provider to answer (parts of) a request is the interaction partner. Though, there are valid practical cases where the data about the interaction partner are to be requested from a third party during an interaction. Take as an example a user whose authorization policies protecting the data of its identity relationship formulae require that the interaction partner have a certain minimum reputation score as stated by a reputation provider the user trusts for the purpose of providing such information. In this case, the request needs to specify information about the subject of the to-be-requested data such that the third party can return the requested data. Such a request is called *third-party request*. Specifying the data subject is done by specifying relevant information in the request, e.g., a jointly known or public identifier of the subject of the request, as in the following example. For subjects with a publicly known identifier (public pseudonym) such as most service providers in today's Internet, the use of such an identifier is the simplest approach as there is no need for an additional agreement of a new identifier between the party and the third party for the subject.

Example 4.22 *Request formula with identity specification* Next, we present a request fragment for a third-party request for confirming a minimum reputation score:

$$\begin{aligned} & \dots R.\text{certifier} = c \dots \\ & \dots \text{Geq}(R.\text{reputationScore}, 8) \wedge \\ & \quad \text{Eq}(R.\text{uniqueName}, \text{"online.Electronics.Shop"}) \dots \end{aligned}$$

The predicate over the *reputationScore* attribute needs to be answered in the response, that is, the given predicate in the example expresses the usual request semantics for attribute information, while the attribute *uniqueName* is provided through its value and used for identifying the party about which the reputation is requested. Using the concept of *sanitizing policies* (see Bonatti and Samarati [BS02] or Ardagna et al. [ACDS08]) that can protect sensitive information contained in policy rules on the side of the requesting party allows for transforming this request into a request for the value of the reputation score before being sent to the third party, thereby hiding the predicate expressed over the attribute value. This prevents the subject expression, the *Geq* predicate and constant 8, of this policy rule of the user from leaking to the third party. In this case, the predicate for requesting the *reputationScore* being greater than or equal to 8 would be replaced by the following predicate where variable *Rep* stands for the attribute value:

$$\text{Eq}(R.\text{reputationScore}, \text{Rep}) . \quad (4.19)$$

Such transformations are relevant in the context of authorization policies based on our data model and (trust) negotiation protocol based on it [Som11].

Architecture. Requests targeted at the interaction partner and asking for data about it or other parties are processed by the interaction partner against its identifier

and identity relationships as well as conditionally released and opaque identities and profiles.

Realizing third-party requests requires that such requests be indicated as such and that the third party can be determined from the request. Those requests are executed against the third party and contribute further information for making an authorization decision. Third-party requests are relevant for a generic system in the context of realizing a negotiation protocol and can be seen as an extension of a basic system.

4.6.5 Authorization Policy Model

The representation of data requests is a main component of our authorization policy language presented by the author in related work [Som11]. Being based on our data model implies that features that can be expressed in data requests become immediately available in the authorization language. This makes it a powerful language which allows, to mention only some crucial aspects, for privacy-enhancing requests while retaining requester accountability and supports delegation over attribute authority.

The author explains in related work [Som11] how the data request language is extended to a full privacy-enhancing authorization language, building on concepts of established work [BS02, ACDS08].

Through the delegation capabilities of our data model, our authorization policy model allows a party to express its policies such that it can allow for a resource both non-delegated and delegated accesses as those two cases are different in terms of identity statements and thus policies. Thus, delegation can be precisely controlled by the relying party with the disadvantage that it learns whether a delegation is in place. Support of the traditional delegation model based on a delegation of rights and handling thereof through unique party identities in the authorization policy is easy to support in the authorization language in addition to our approach.

4.7 Repository and Portfolio

As briefly outlined in Sec. 4.3, a party holds a *repository* of data and a *portfolio*. The repository comprises *identifier relationships*, *identity relationships*, the *data track*, and the *profile data* of a party. The *portfolio* comprises a subset of the repository and is used for matching data a party holds against policies to fulfill. For third-party requests the profile data is considered in addition to what is needed for regular requests about the party. In this section, we discuss the technical representation of data by a party in our system and purposes of use of the data.

4.7.1 Identifier Relationships

The identifier relationships \mathcal{R}^{idf} of a party represent *identifiers*, or *pseudonyms*, a party has established, each one being about a single party—be it the party itself or

another party, e.g., a delegatee—for use in communication with another party or multiple other parties. The set \mathcal{R}^{idf} is a set of identifier relationships ϵ_i^{idf} , $1 \leq i \leq \ell^{\mathcal{R}^{\text{idf}}}$ comprising both the data and metadata of the identifier relationship where the data are represented using our data model as outlined in this subsection. Thereby, $\ell^{\mathcal{R}^{\text{idf}}}$ refers to the number of identifier relationships of the party at a given time, that is, in a given state of the system.

An identifier relationship ϵ^{idf} is a tuple comprising an element $\hat{\epsilon}^{\text{idf}}$ and a set of metadata predicates associated with the element. The element is a set of cardinality 1 of a tuple $\hat{\epsilon}^{\text{idf}}$ comprising a formula ϕ specifying an *identifier object* \mathbf{p} through its holder, subject, and the subject identifier value and a set of metadata predicates $\{\diamond_j^{\text{idf}}\}$.²⁷

The formulae used for expressing the attributes of an identifier object in an identifier relationship and the formulae representing the input and output to the protocols for establishing identifier relationships are elements of the formal language \mathcal{L}^{idf} . This language is a small fragment of what our data representation is capable of expressing, focused on pseudonym-related aspects thereof. We introduce the language informally in the following, as part of the explanation of identifier relationships and the protocol interface.

\mathcal{L}^{idf} can be characterized as the language comprising only Eq predicates expressed on attribute references of identifier objects, variables representing attribute values, and constant attribute values as their arguments, connected only through the \wedge -connective. The identifier objects may either be denoted through variable or constant terms. The language \mathcal{L}^{idf} is the basis for the languages $\mathcal{L}^{\text{idf}/\mathbf{p}}$ for specifying the protocol for establishing an identifier relationship and $\mathcal{L}^{\text{idf}/\mathbf{r}}$ for representing an identifier relationship at its holding party once established. The relations $\mathcal{L}^{\text{idf}} \subset \mathcal{L}^{\mathbf{g}}$, $\mathcal{L}^{\text{idf}/\mathbf{p}} \subset \mathcal{L}^{\text{idf}}$, and $\mathcal{L}^{\text{idf}/\mathbf{r}} \subset \mathcal{L}^{\text{idf}}$ hold for the languages.

The basic structure of a formula $\phi \in \mathcal{L}^{\text{idf}/\mathbf{r}}$ representing an identifier relationship is presented in (4.20)—see Sec. 4.4.7 for the concepts. Such formula is henceforth also referred to as *identifier object specification formula*. The formula makes a statement about the attributes of the identifier object \mathbf{p} of the identifier relationship. The symbols \mathbf{h} and \mathbf{s} refer to concrete pty -typed constants specifying the *holder* and *subject*, respectively, and sid to the pseudonym identifier of type idf . In an interpretation of the formula, the values for the holder and subject will be interpreted with the party, the *subjectId* with itself. The language $\mathcal{L}^{\text{idf}/\mathbf{r}}$ is, in contrast to \mathcal{L}^{idf} and $\mathcal{L}^{\text{idf}/\mathbf{p}}$, characterized by comprising only constant terms.

$$\phi = \mathbf{p}.\text{holder} = \mathbf{h} \wedge \mathbf{p}.\text{subject} = \mathbf{s} \wedge \text{Eq}(\mathbf{p}.\text{subjectId}, \text{sid}) \quad (4.20)$$

An identifier relationship ϵ^{idf} contains a metadata predicate specifying the subject term, that is, a reference to the party to whom the identifier relationship applies. The following metadata are associated with an identifier object specification formula:

²⁷The identifier relationship is modeled with a set of tuples of formulae and metadata with a restriction that the set be of cardinality 1 in order to use the same modeling as for the other kinds of data. The restriction on the set applies as for identifier relationships only one formula is required for its modeling.

the subject identifier of the party with whom the relationship has been established, the mapping from the locally used terms to the terms used in communication with the other party, cryptographic material (e.g., a cryptographic pseudonym) for using the identifier relationship, and further metadata. An identifier relationship can be obtained by the party by executing an instance of the `EstIdtfRel` or `EstIdtfRelReg` protocol with another party.

4.7.2 Identity Relationships

An identity relationship captures the decision of a *certifier* to vouch for data about a party, also referred to as *certified*. A party holds a set \mathcal{R}^{idy} of identity relationships. An identity relationship ϵ^{idy} is a tuple of an element $\hat{\epsilon}^{\text{idy}}$ and a set of metadata predicates on $\hat{\epsilon}^{\text{idy}}$. The element $\hat{\epsilon}^{\text{idy}}$ is a set of tuples $\tilde{\epsilon}^{\text{idy}}$ each comprising a formula ϕ_i and a set of metadata predicates on the formula. Let $\epsilon_1^{\text{idy}}, \dots, \epsilon_{\ell^{\text{idy}}}^{\text{idy}}$ be the identity relationships of a party at a given time.

A strength of the modeling of certified identity data through identity relationships is their embedding into the overall formalism and expressivity provided for by our data model and thus integration into our architecture. Our approach particularly allows for general ways of associating authorization policies with the data of the identity relationship or selected parts of it as explained in earlier work [Som11].

4.7.2.1 Formulae for Specifying Identities

The key element of an identity relationship is an *identity* which is described by the identity relationship and also referred to as *base identity*. This is the identity comprising the attributes the identity relationship is about—additional identities, typically one, may be used by a party to specify the certifier of the base identity. A formula $\phi \in \mathcal{L}^{\text{idy}/r}$, henceforth denoted as *identity specification formula*, expresses the identity data of the base identity c and its in-formula metadata in an integrated way through our data representation approach. Thereby, it specifies exactly the data that the certifier of the identity relationship vouches for and specifies the certifier of the base identity. The permitted syntax for $\mathcal{L}^{\text{idy}/r}$ is governed by the protocol underlying the identity relationship—our discussion will cover private certificate systems.

We specify the language $\mathcal{L}^{\text{idy}/r}$ informally as a fragment of \mathcal{L}^{E} comprising only \wedge -connected predicates. A formula ϕ in $\mathcal{L}^{\text{idy}/r}$ specifies the holder and subject through a predicate each, relating the *holder* and *subject* attributes with a corresponding constant term through the $=$ -relation. The *certifier* attribute is related through the $=$ -symbol to a constant term referring to the certifier of the base identity. No further predicates for specifying the certifier are required—it is specified outside of the identity specification formula. The *type* and *protocol* attributes are related to constants through `Eq` predicates. For specifying the user attributes of ϕ , the formula expresses `Eq` predicates over attributes of the base identity, relating them with constants or—for the extension of restrictions as explained below—variables,

which is the only permitted use of variables here. $\mathcal{L}^{\text{id}/r}$ is a fragment of the language \mathcal{L}^{id} introduced in Sec. 5.3.

Certifier specification. In addition to a formula ϕ , the party holds—or obtains in a sub-protocol or protocol preceding that for establishing the identity relationship—another formula $\hat{\phi}$ that specifies the certifier of the base identity by making statements about a certifier identity cid . The certifier identity is associated with the base identity as explained in Sec. 4.4.10 through a term of type **pty** being used as value of the *certifier* attribute of the base identity and *subject* attribute of the corresponding certifier identity. The formula $\hat{\phi}$ is stored by the party as part of an entry of the party’s *profile data*, flagged as certifier data related to the certifier of one of the party’s identities. The subject of the certifier’s identity is the certifier of the base identity. When multiple identities of identity relationships have the same party as their certifier referred to through the same subject term and specified through the same certifier identity, the certifier formula for the certifier needs to be specified only once in the profile data of the party as it always describes the same identity of the same certifier. The profile data entry based on the certifier’s identity cid results from the certifier having an identity relationship with cid corresponding to a public key certificate. See Sec. 4.8 for further details on how certification of keys based on traditional RSA [RSA78, RSA83], DSA [Nat09], or other technologies is modeled. Note that $\hat{\phi}$ can also make statements about an identifier object, or multiple identities or identifier objects of the certifier in a more generic way.

In the most prominent case of an identified certifier that acts under a public pseudonym, the formula $\hat{\phi}$ comprises identifying attribute statements about one certifier’s identifier object or identity or both and the subject term for referring to the certifier is a publicly used term corresponding to its public pseudonym used by all parties for referring to this certifier. For the less prominent case of the certifier remaining pseudonymous, it can appear under a pseudonym and attribute statement to which its credential verification public key is bound.

Multiple data formulae and restrictions. Multiple formulae ϕ_1, \dots, ϕ_l can be associated with the same identity relationship ϵ^{id} . One formula, denoted as the *core formula* and tagged through the metadata predicate $\Diamond(\text{“coreFormula”, true})$ as such, must specify precisely what the certifier vouches for. Let, without loss of generality, be ϕ_1 the core formula. Additional formulae may be used to specify a fragment of the data of the identity relationship, that is, a fragment of the data the certifier vouches for. This can be used for associating a different authorization policy with such fragment [Som11]. Each of the other formulae must be derivable from the core formula in the logic calculus, that is, it must hold for all ϕ_i that $\phi_1 \vdash \phi_i$ for $2 \leq i \leq l$. A use case for this is that the core formula specifies a value for a specific attribute of the identity while an additional formula ϕ_2 specifies one or more predicates over the attribute. Then, a restrictive authorization policy can be associated for the release of the attribute value in ϕ_1 while a less restrictive authorization policy can be defined on the predicates over the attribute in ϕ_2 , e.g., for responding to age

verification requests. We next introduce the concept of *restrictions* for more powerful association of authorization policies governing the release of data based on an example. Restrictions are an extension to our system and require a more complex language for specifying identity specification formulae due to the use of variables.

Example 4.23 (Multiple formulae and restrictions) *With this example we show the use of an additional formula of an identity relationship and introduce the concept of a restriction. Let the formula ϕ_2 define a predicate on the attribute `monthlySalary` of the salary statement identity `b` expressing that the attribute be greater than or equal to the variable `Lowerbound`. Let furthermore the restriction ρ_{ϕ_2} on ϕ_2 specify predicates on the range of this variable.*

$$\begin{aligned}\phi_2 &= \text{Geq}(\text{b.monthlySalary}, \text{Lowerbound}) \\ \rho_{\phi_2} &= \text{Leq}(2500, \text{Lowerbound}) \wedge \text{Leq}(\text{Lowerbound}, 4000)\end{aligned}$$

The above example formula ϕ_2 and its restriction ρ_{ϕ_2} define that the `salary` attribute of the identity be greater than or equal to the variable `Lowerbound`, where the latter ranges—specified through the restriction ρ_{ϕ_2} —from 2500 to 4000, both inclusive. A different, e.g., less restrictive authorization policy than the one on the core formula ϕ_1 , as ϕ_1 specifies a concrete value for the attribute `monthlySalary`, can be defined on the formula ϕ_2 , because it reveals only a predicate over the attribute and thus much less information.

Matching formula ϕ_2 with its restriction against a data request using reasoning in our logic needs to consider the formula ϕ_2 as well as its associated restriction formula ρ_{ϕ_2} . Both need to be true under the chosen assignment of the variable `Lowerbound` and considered in the choice of values in the matching algorithm. The restriction never becomes part of the response to a data request, but is only used for finding suitable responses.

Example 4.24 (Multiple formulae) *As a special case of the previous example, the following formula specifies the release of partial information on the attribute using a constant reference value and without the flexibility of using a restriction. Again, authorization policies governing the release of this predicate in negotiation interactions can be associated with the predicate, see Sommer [Som11] for details.*

$$\phi_3 = \text{Geq}(\text{b.salary}, 3000)$$

The approach of specifying ranges to be revealed fits well into our formalism for data representation without the need of extensions. Authorization policies and data handling policies can be associated with each formula ϕ_* of the identity relationship as well as their predicates.

In addition to ranges over the integers as explained above or other attributes over totally ordered sets, a use case for multiple formulae in identity relationships is that the country of residence attribute of an electronic identity credential be released to be in the set of all EU and EFTA member states without restrictions on the use of it. For use cases, though, where the value of the country of residence is to be revealed, a

stricter authorization policy or data handling policy may be required to be enforced, e.g., the potential recipient would need to prove, in a negotiation protocol, that it holds a certification by one of a set of data protection organizations of applying a minimum standard regarding its privacy practices. Note that set membership $a \in S = \{a_1, \dots, a_l\}$ can be expressed by a disjunction $\text{Eq}(a, a_1) \vee \dots \vee \text{Eq}(a, a_l)$ for all $a \in S$ in our first-order logic.

4.7.2.2 Metadata

An identity relationship and its formulae require metadata to be associated with them in order to be utilized for releasing data in interactions with other parties through instances of the RelData protocol. We give an overview of the most important metadata of identity relationships and its formulae next.

The metadata comprises cryptographic material, such as private certificates or public keys, in case they are stored directly within the identity relationship. We also need a metadata predicate that specifies whether an identity relationship is one the party is holder of or whether it is one the party vouches for. If we do not specify this explicitly in this work, we assume that it is clear from the context. Another predicate expresses whether the identity relationship is still active or has been deactivated, e.g., because it has expired, been revoked, or been superseded due to a change of attribute data. The core formula of the identity relationship is flagged through a metadata predicate as such. The metadata must express everything that is (technically) needed for the identity relationship to be utilized in interactions for releasing data and that is not contained yet in the data formula. For an identity relationship based on a private certificate, this particularly comprises the *certificate structure*, a technical specification of the internals of private certificates of a specific type. We present the notation for metadata through metadata predicates in Sec. 5.2.

4.7.3 Data Track

The data track $\mathcal{R}^{\text{trck}}$ of a party models data statements that have been released by the party to other parties as well as associated metadata. The idea is that each release of identity-related information that the party's system is aware of is recorded in the data track. Of our interest are particularly all release transactions through instances of the RelData protocol.

A party's data track $\mathcal{R}^{\text{trck}}$ is a set of tuples, denoted data track entries or data track records, where each tuple ϵ^{trck} comprises an element $\hat{\epsilon}^{\text{trck}}$ and a set of metadata predicates on this element. Each such element $\hat{\epsilon}^{\text{trck}}$ comprises a set of tuples $\tilde{\epsilon}^{\text{trck}}$ of the form $(\varphi_k^{\text{trck}}, \{\diamond_{\varphi_k^{\text{trck}}, I}\})$ of a formula φ_k^{trck} expressed in a technology-dependent fragment of $\mathcal{L}^{\text{p/g}}$ and a set of metadata predicates associated with the formula.

A single data track record ϵ^{trck} comprises data related to a single recipient. Multiple data releases by the party within a single session and even within multiple sessions with the same recipient can be captured by one record. Formulae in one data track record can refer to multiple subjects, e.g., the party, delegaters, and

certifiers, though, and each record is associated with the main subject the data is about.

A formula represents a data statement that has been released to, through an instance of the protocol *RelData*, or obtained by another party, expressed in \mathcal{L}^p/g . Metadata can be expressed, like for other kinds of data, on the data track record as well as on each formula of a record in order to store relevant information.

4.7.3.1 Data Formula

Each formula of a data track record is a formula that has been previously released to or otherwise obtained by another party that has been captured in the data track after the data release interaction. The formula contains the identity data as well as metadata sent to the other party.

The formula stored as part of a data track entry is that after applying the term renaming to the formula the party has derived from its portfolio as explained in detail in Sec. 4.8. That is, exactly the formula as released to the other party gets stored. This immediately shows linkabilities that are explicitly established through the used terms between different interactions with the other party or other parties. As the mapping between local and released terms for the formula is available to the party, it can always obtain the terms it uses itself for addressing parties and objects.

4.7.3.2 Metadata

Metadata on a data track element \hat{e}^{trk} comprises a term specifying the recipient of the data, that is, the party whom the record is associated with. The convention for the choice of an identifier of the recipient is the *subject* term of the party the formula has been released to. For each such subject there exists a profile data record with the same identifier which may contain information about the other party data has been released to, unless in the case that data has been released to a party that has not provided any data about itself.²⁸

The subject of the released data is already represented in the formula specifying the data and is, for faster access, also stored as a metadata predicate. Note that the subject is often the party itself, but can (additionally) be another party in case the party releases data about other parties, e.g., delegaters, as may be the case for a service provider who releases data of their customers to a business partner or a user who acts for another party under a delegation relationship.

Metadata predicates expressed on each data formula of the data track record comprise the following: the data handling policy under which the data has been released (through its identifier or copy of the policy), identity and identifier relationships and profiles that the data statement is based on, expressed as references

²⁸This case is prominent for a service provider who releases data about itself to anonymous users who have not (yet) provided data about themselves. For commercial service providers it is practically not interesting to track to whom data about their legal identity has been released to.

to those and their formulae, the date and time of release, the subject of the released data, the proof used in the data exchange protocol, annotations by the party (e.g., the user), the session identifier, and further metadata required for bookkeeping. Particularly important metadata are the cryptographic objects corresponding to the opaque and conditionally released identities generated in the instance of the protocol for releasing data.

Relating a formula in the data track to the data handling policy it has been released under allows the party to later assess in an interaction with the other party the enforcement state of the data handling policy and, in case of issues, take further actions.²⁹ Annotations by the user may comprise comments, e.g., to distinguish multiple partial identities she has with the other party. Work on those aspects of user-centric identity management have been done within the PRIME project [PRI08, CLS11]. In the current work, we integrate those aspects tightly with the privacy-enhanced authentication mechanism and formalism for data representation.

4.7.4 Profile Data

The profile data \mathcal{R}^{pro} of a party comprises a set of tuples, each comprising a profile record, and a set of metadata predicates of the form $(\tilde{c}_i^{\text{pro}}, \diamond_{\tilde{c}_i^{\text{pro}}, j})$. A profile record \tilde{c}_i^{pro} is a set of tuples $(\varphi_k^{\text{pro}}, \{\diamond_{\varphi_k^{\text{pro}}, l}\})$, each comprising a formula and a set of metadata predicates. Note that this structure is the same as the structure of the other kinds of data. A term referring to the subject for the party a record \tilde{c}_i^{pro} applies to is associated with the record through a metadata predicate.

Whenever a party obtains data about another party, e.g., by receiving them from the other party, from third parties, or through any other means, those data are stored in the profile record about the other party, thereby creating an identity profile of the other party. A user creates a profile for each service provider she interacts with to store the information she has obtained about the service provider. As is done for all classes of data, such data are expressed through formulae. This is at least the information stored in the service provider's public key certificate, such as its distinguished name, country, and Uniform Resource Locator (URL) of the service. During a negotiation, the user may obtain further data, e.g., about the reputation, assurance mechanisms, or trust assessment of the service provider [CLS11]. A service provider creates, for such interaction, a profile data entry for each customer it interacts with comprising the data received. When a customer uses transaction pseudonyms, each time a new profile data entry is created about it by the service provider. This approach reflects exactly the use of pseudonyms by a party.

A crucial class of profile data entries are those related to certifiers of identity relationships the party is holder of and are, as explained earlier, flagged as such entries. Those entries are required for the matching of a party's portfolio against a policy.

Profile data needs to, regardless of its potential of identifying the provider of the data, be handled according to the data handling policies agreed with the provider.

²⁹One can think of an electronic assistant for filing a complaint with the data protection authority responsible for the data processing.

Following European data protection legislation, users do not need to enforce any policy on identity data received from a service provider as this is considered a so-called household activity, the case of data received from (about) other users is different and may impose the legal responsibility of being data controller on a user.

4.7.4.1 Data Formula

The identity data of profile entries is represented through formulae expressed in our data model. Valid formulae are those that comply with the syntax of a fragment of the data statement language $\mathcal{L}^{p/g}$ corresponding to the employed (cryptographic) mechanism for releasing (certified) data.

4.7.4.2 Metadata

One metadata predicate on a profile record is the subject term of the party the profile is about. All formulae received from or about the other party under the same subject term can be stored under the same profile record.

For a formula, the following are examples of metadata items to be stored: date and time of reception of the data, party identifier whom the data have been received from, protocol transcript of the data release protocol including all cryptographic tokens, and other metadata required for bookkeeping. An important metadata predicate is the one indicating whether the party the entry is about is a certifier of an identity of an identity relationship of the party as previously discussed. Based on this, a subset of entries is retrieved for matching the party's repository against a policy as explained in Sec. 5.4.

4.7.4.3 Use

Profile data has multiple uses in our architecture. When evaluating an access request to a resource of the party, profile data of the other party may be used to supply data for the authorization decision in addition to the data obtained within the ongoing transaction. This is possible only in case the other party is already known under at least a subject term. When a user browses her data track, parts of the profile data of the data recipients can be mapped into the view the user is presented with in order to provide an enhanced user experience when assessing her releases of data. For a user's legal right under European legislation of access to data [Eur95], her profile about the other party specifies relevant information on how to access data at the other party, for example, the URL under which this service is provided.

4.7.5 Implementation

An implementation of the repository (and portfolio) can be done following standards, e.g., through a (relational) database. It is crucial that frequently used search and retrieval operations be realized sufficiently fast, e.g., through indexing or index relations. Our system is agnostic to the concrete technology employed for implementing the repository.

4.8 Applying the Data Model

We next present an overview on how to apply the data representation language for two main purposes: (1) establishing identifier and identity relationships of a party and establishing the local representation of corresponding specification formulae, and (2) representing formulae to be released to data recipient parties, that is, data statements, based on those identifier relationships and identity relationships. We note that formulae derived in the logic calculus from formulae released to a party through cryptographic protocols can be less constrained than data statements.

In this section, we first cover aspects related to (1) in Sec. 4.8.1 and aspects related to (2) in Sec. 4.8.2 in detail for pseudonym and credential protocols and thereafter give a brief account of those aspects for the simpler setting of conventional certificates based on RSA, DSA, ECDSA, or other conventional signature algorithms in Sec. 4.8.3.

4.8.1 Establishing Objects and their Local Representations

A party needs to represent its identifier and identity relationships locally, particularly the formulae specifying the underlying identifier objects and identities.³⁰

Note that, for the below discussions, there is a one-to-one correspondance between identifier objects and identifier relationships, and identities and identity relationships and we often use the corresponding terms interchangeably.

4.8.1.1 Identifier Objects

Recall that an identifier object p_i and its object specification formula ϕ_i is established as the core part of an identifier relationship ϵ_i^{idf} of the party with another party. An identifier object is created through the protocol `EstIdtfRel` or `EstIdtfRelReg` for establishing an identifier relationship and is discussed in Chapter 5. In the setting of private certificate protocols, an identifier object is technically realized through a corresponding *cryptographic pseudonym*.

We need to differentiate between registration identifier objects and non-registration identifier objects and the corresponding identifier relationships in one dimension, and whether the identifier is established for a party itself or a delegatee in the other dimension, resulting in four cases in total. Those cases expose differences in terms of the terms of the logic used for referring to the *holder* and *subject* attributes of the established objects in the object specification formulae local to the party and the term renaming between local representation and use of terms in data statements to be disclosed.

During the establishment protocol for an identifier object, the *holder*, *subject*, and *subjectId* terms for the identifier object are created based on the underlying (cryptographic) protocol, though, may be renamed to different terms used in the

³⁰Note that conditionally released identities and opaque identities need to be represented as well in our system, though, are of less interest in the current context because their attributes are not related to *holder* and *subject* terms.

specification formula for the created object locally to facilitate deductions in the logic, depending on the case. When using the object in a data statement to be released, the terms may be renamed back to the ones created for the particular object.

Registration identifier object. \mathbf{A} establishes a registration identifier object $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$ with itself being the subject and holder with party \mathbf{C}_i to perform the first step of a registration in the system. $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$ is part of the identifier relationship $\epsilon_{A, \mathbf{C}_i, \ell_r}^{\text{idf}}$ with object specification formula $\phi_{A, \mathbf{C}_i, \ell_r}$ specifying $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$. The term $\mathbf{t}_{A, \mathbf{C}_i, \ell_r}$ is generated through the protocol for referring to the holder and subject of the identifier object. This term is used in the local data representation for specifying the subject and holder of $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$ in $\phi_{A, \mathbf{C}_i, \ell_r}$, as well as when making statements about the subject and holder towards \mathbf{C}_i .³¹ The term mapping function $\varrho_{\mathbf{p}_{A, \mathbf{C}_i, \ell_r}}^{\text{idf}}$ is the empty function.

Each registration identifier object $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$ induces a *registration domain* referred to as $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \ell_r}}$. A registration domain comprises all identifier objects and identities which are associated with the same party \mathbf{A} holding $\mathbf{p}_{A, \mathbf{C}_i, \ell_r}$ and in the party's local object specification formulae the holder is referred to with the same term $\mathbf{t}_{A, \mathbf{C}_i, \ell_r}$. For objects with the party \mathbf{A} as subject, also the subject is referred to with this term. Using credential protocols, each such registration domain corresponds exactly to a private key domain of the party—see Sec. 5.6. Two registration domains can be bridged in the semantics of the data model by adding the relation $\mathbf{t}_i = \mathbf{t}_j$ to the party's portfolio, where the terms \mathbf{t}_i and \mathbf{t}_j are the terms used for referring to the party in the two domains. This bridging has the semantics that both (different) terms in the relation refer to the same party.

Derived identifier object. \mathbf{A} establishes a non-registration, or derived, identifier object $\mathbf{p}_{A, \mathbf{C}'_j, \ell_o}$ with itself being the subject and holder with party \mathbf{C}'_j . The object is established with respect to a registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \ell_r}}$. The object $\mathbf{p}_{A, \mathbf{C}'_j, \ell_o}$ is part of the identifier relationship $\epsilon_{A, \mathbf{C}'_j, \ell_o}^{\text{idf}}$ with specification formula $\phi_{A, \mathbf{C}'_j, \ell_o}$. For such non-registration, or derived, identifier object, the term for its holder and subject attribute values used locally in $\phi_{A, \mathbf{C}'_j, \ell_o}$ and data statements before term renaming is $\mathbf{t}_{A, \mathbf{C}_i, \ell_r}$, the term used in data statement formulae to be released to \mathbf{C}'_j for both functions is $\mathbf{t}_{A, \mathbf{C}'_j, \ell_o}^{h,s}$, generated as part of the establishment protocol of the object. The terms used in data statements after term renaming are different to the corresponding terms of the registration identifier object in the registration domain of which $\mathbf{p}_{A, \mathbf{C}'_j, \ell_o}$ is contained. The term mapping function $\varrho_{\mathbf{p}_{A, \mathbf{C}'_j, \ell_o}}^{\text{idf}}$ corresponding to $\mathbf{p}_{A, \mathbf{C}'_j, \ell_o}$ expresses the mapping between \mathbf{A} 's local representation and the terms to be used in communication with other parties.

³¹Those terms are not released in data statements to any other party than \mathbf{C}_i , except for as values of the *subjectIdWithCertifier* attribute of identities.

Delegation registration identifier object. \mathbf{A} establishes a delegation registration identifier object $\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}$ with itself being the holder and a delegater \mathbf{D} being the subject with a delegation certifier \mathbf{C}_k^* . The object is established with respect to a registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \ell_r}}$ of \mathbf{A} . $\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}$ is part of the identifier relationship $\epsilon_{A, \mathbf{C}_k^*, \ell_d}^{\text{idf}}$ with specification formula $\phi_{A, \mathbf{C}_k^*, \ell_d}$. Such identifier object is established by a delegatee \mathbf{A} with a delegation certifier \mathbf{C}_k^* before establishing a delegated identity with party \mathbf{D} as the subject. The terms used locally for the holder and subject in $\phi_{A, \mathbf{C}_k^*, \ell_d}$ and data statements before term renaming are $\mathbf{t}_{A, \mathbf{C}_i, \ell_r}$ and $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^s$, respectively. The latter is used to refer to \mathbf{D} in the local data representation of \mathbf{A} . The terms used when making statements about the object towards \mathbf{C}_k^* are $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^h$ and $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^s$, respectively.³² Both those terms are created during the establishment of the identifier object. The mapping function $\varrho_{\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}}^{\text{idf}}$ specifies the mapping between the local terms used by \mathbf{A} and terms to be used with other parties.

A delegation registration identifier object $\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}$ induces a *delegation registration domain* $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}}^{\mathbf{D}}$, analogous to a registration domain. The domain comprises $\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}$ and all delegation identifier objects and delegation identities established with \mathbf{D} being the subject, and being referred to by \mathbf{A} locally through $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^s$.

Delegation identifier object. \mathbf{A} establishes a delegation identifier object denoted as $\mathbf{p}_{A, D, B, \ell_b}$ with itself being the holder and the delegater \mathbf{D} being the subject with party \mathbf{B} . The object is established with respect to a registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \ell_r}}$ of \mathbf{A} and a delegation registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_k^*, \ell_d}}^{\mathbf{D}}$ of \mathbf{A} about delegater \mathbf{D} . $\mathbf{p}_{A, D, B, \ell_b}$ is part of the identifier relationship $\epsilon_{A, D, B, \ell_b}^{\text{idf}}$ with specification formula ϕ_{A, D, B, ℓ_b} . The holder and subject terms used locally are $\mathbf{t}_{A, \mathbf{C}_i, \ell_r}$ and $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^s$, the terms used in data statements are the terms $\mathbf{t}_{A, D, B, \ell_b}^h$ and $\mathbf{t}_{A, D, B, \ell_b}^s$, respectively, created together with the identifier object. The renaming function $\varrho_{\mathbf{p}_{A, D, B, \ell_b}}^{\text{idf}}$ specifies the mapping between \mathbf{A} 's locally used terms and those used with remotely parties.

Discussion. For the local representation, a party always uses, as holder and subject terms for its identifier objects and identities, the terms created when establishing the registration identifier object in the registration domain of which those identifier objects and identities are contained in. For underlying cryptographic objects created through private certificate protocols, this means that the same user private key that is created for the establishment of the cryptographic pseudonym underlying the registration identifier object is used for the cryptographic pseudonyms underlying the identifier objects and the private certificates underlying the identities in the same registration domain.

For an identifier object that a party establishes having another party (delegater) as the subject, analogous cases of registration and non-registration identifier objects

³²Note that, analogous to a registration identifier, $\mathbf{t}_{A, \mathbf{C}_k^*, \ell_d}^s$ is only used towards \mathbf{C}_k^* .

apply as for identifier objects with the party itself being the subject. Creating a delegation registration identifier object is done by \mathbf{A} for establishing an identifier relation with a delegation certifier \mathbf{C}_k^* .³³ The holder term is handled as in the case of identifier objects with the party itself being the subject, while the subject term is handled analogously. The subject term is different to the holder term and refers to the delegator. For a delegation identifier object, the subject term is created during the establishment of the object and renamed, for local representation, to the subject term generated in the protocol for establishing the delegation registration identifier object in the delegation registration domain of which the newly generated object resides. This again implements the idea of using the same term for the subject throughout the registration domain locally, analogous to identifier objects with the party as subject.

Note that the subject identifier $subjectId :: idf$ of any identifier object is never renamed to other terms, because it is the fixed pseudonym corresponding to the identifier object. Also, the term p for referring to the identifier object itself is, by convention, not renamed when using it, because the purpose of an identifier object is to establish linkability between the transaction of its establishment and all of the transactions in which it is referred. Both this holds for all kinds of identifier objects discussed above.

Once established, an identifier relationship and its identifier object can be used for proving holdship of the object, i.e., the underlying pseudonym, to any of the parties it has been established with, as part of a data release protocol. The subject and holder terms of the identifier object can be used also as subject and holder terms for identities referred to in the same formula to be released.

Derived and delegation identifier objects can have the property of being domain identifier objects, that is, \mathbf{A} can only establish a single identifier object with each data recipient for one domain string.

4.8.1.2 Identities

An identity relationship ϵ_i^{idy} comprises, as its core part, an object specification formula φ_i^{idy} specifying an identity c_i . It is established through the protocol discussed in Chapter 5. An identity relationship is established with respect to identifier objects which specify its holder and subject. An established identity relationship is used for releasing data specified through its associated object specification formula to another party.

Registration identity. A party \mathbf{A} can obtain a *registration identity* $c_{A, \mathbf{C}_i, \hat{\ell}_r}$ with itself as holder and subject from a registration certifier \mathbf{C}_i part of the registration domain of a registration identifier object $p_{A, \mathbf{C}_i, \ell_r}$. The subject and holder term used to refer to \mathbf{A} as the holder and subject of the identity locally is $t_{A, \mathbf{C}_i, \ell_r}$ related to the registration identifier object $p_{A, \mathbf{C}_i, \ell_r}$. The identity corresponds to an identity

³³For our protocols based on credential systems, this delegation registration identifier object corresponds to a new private delegation key of \mathbf{A} .

relationship $\epsilon_{A, \mathbf{C}_i, \hat{\ell}_r}^{\text{idy}}$ with its specification formula $\phi_{A, \mathbf{C}_i, \hat{\ell}_r}$ specifying $\mathbf{c}_{A, \mathbf{C}_i, \hat{\ell}_r}$. The identity is part of the registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}}$.

Non-registration identity. A party **A** can obtain a *non-registration identity*, or *derived identity*, $\mathbf{c}_{A, \mathbf{C}'_j, \hat{\ell}_o}$ with itself as holder and subject from a certifier \mathbf{C}'_j with respect to a derived identifier object $\mathbf{p}_{A, \mathbf{C}'_j, \hat{\ell}_o}$ part of $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}}$. The subject and holder term used to refer to **A** as the holder and subject of the identity locally is $\mathbf{t}_{A, \mathbf{C}_i, \hat{\ell}_r}$ as for the registration identity $\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}$. The identity $\mathbf{c}_{A, \mathbf{C}'_j, \hat{\ell}_o}$ corresponds to an identity relationship $\epsilon_{A, \mathbf{C}'_j, \hat{\ell}_o}^{\text{idy}}$ with its specification formula $\phi_{A, \mathbf{C}'_j, \hat{\ell}_o}$ specifying $\mathbf{c}_{A, \mathbf{C}'_j, \hat{\ell}_o}$. The identity is also part of the registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}}$.

Delegation identity. A party **A** can obtain a *delegation identity* $\mathbf{c}_{A, \mathbf{C}_k^*, \hat{\ell}_d}$ with itself as holder and a delegater **D** as subject from a certifier \mathbf{C}_k^* with respect to a registration or derived identifier object of **A** that is part of $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}}$ specifying the holder and a delegation registration identifier object $\mathbf{p}_{A, \mathbf{C}_k^*, \hat{\ell}_d} \in \mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_k^*, \hat{\ell}_d}}$ determining the subject **D** of the identity. The term used to refer to **A** as the holder of the identity locally is $\mathbf{t}_{A, \mathbf{C}_i, \hat{\ell}_r}$ as for the registration or derived identifier object, and the subject term for specifying the delegater **D** is $\mathbf{t}_{A, \mathbf{C}_k^*, \hat{\ell}_d}^s$. The identity $\mathbf{c}_{A, \mathbf{C}_k^*, \hat{\ell}_d}$ is part of the registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_i, \hat{\ell}_r}}$ and delegation registration domain $\mathcal{R}_{\mathbf{p}_{A, \mathbf{C}_k^*, \hat{\ell}_d}}$, that is, the domains the identifier objects it is based on are part of. The identity corresponds to identity relationship $\epsilon_{A, \mathbf{C}_k^*, \hat{\ell}_d}^{\text{idy}}$ with object specification formula $\phi_{A, \mathbf{C}_k^*, \hat{\ell}_d}$.

Discussion. To summarize, the subject and holder attributes used in a formula specifying an identity relationship in a party's portfolio are those of the formulae specifying the identifier objects to which the identity relationship is associated, that is, those which specify its holder and subject. This approach ensures that the holder and subject terms used in the local formula for specifying the identity are the ones used for the registration domains of the registration or delegation registration identifier objects for the holder and the subject, respectively.

The underlying approach that a party always uses the holder and subject terms created with registration or delegation registration identifier objects as the ones to express the holder and subject of other identifier objects and identities in the registration domain is crucial for the utility of the resulting system in terms of being required for the automated deduction system being able to establish according relations between different objects. The term renaming for obtaining the local representation of formulae in a party's identifier and identity relationships allows for certain functionality with respect to the reasoning over formulae in our logic calculus, e.g., for determining fulfilling formulae for a non-trivial data request. In other words, this enables the party to use syntactic derivations over our calculus to derive, from a party's formulae in its repository, formulae that can be proven with cryptographic credential protocols.

If the term renaming were not followed and different, even though valid, terms were used by the party to refer to the subject and holder of its identity relationships, this would lead to a situation where either fewer statements that are satisfied can be derived through reasoning over portfolio formulae and thus fewer policies could be fulfilled, or derivable statements could not be proven with the underlying protocols. Though, the semantics in such case were correct as it models the actual situation and not the limitations of the (cryptographic) methods used to implement identity relationships. Particularly, the term (re)naming realizes that the policy matching process works for private certificate schemes for proving that the holder or subject of multiple identities backed by private certificates are the same party.

Each protocol instance for establishing an identifier object or identity establishes a corresponding identifier relationship or identity relationship, together with a corresponding object specification formula.

4.8.2 Data Release

When a party **A** intends to release a data statement ϕ' to another party **B**, it constructs such statement based on the formulae ϕ_1, \dots, ϕ_k of a subset of size k of its identifier and identity relationships as well as conditionally released and opaque identities. Each $\phi_i, 1 \leq i \leq k$ corresponds precisely to one such cryptographic object \mathbf{o}_i . A formula ϕ' of a fragment of $\mathcal{L}^{\mathbf{p}/\mathbf{g}}$ comprises predicates expressed over identifier objects, identities, conditionally released identities, and opaque identities, connected through the connectives of the logic. Note that we simplify notation with respect to the indices used for elements we refer to, in contrast to the notation of Sec. 4.8.1 above.

The formula ϕ' is typically constructed by **A** to fulfill a data request ψ of a data recipient, or relying party, **B**. Thereby, **A** uses at least one registration or derived identifier object as identifier for itself, and delegation registration or delegation identifier object as identifier for a delegater in case of making statements about such party. Statements about multiple delegaters can be made, though this is of less practical interest. Registration identities and derived identities are used for making attribute statements about **A** itself, while delegation identities are used to make attribute statements about delegaters. How all those objects relate to each other has been sufficiently discussed further above and applies here as well due to ϕ' being derived syntactically from the formulae ϕ_i in our logic calculus \mathcal{L} as introduced in Sec. 4.9.2.

We require that for each registration domain of **A**, of which objects are referred in ϕ' , at least one identifier object of the same domain be referred to in ϕ' , and that for each delegation registration domain from which cryptographic objects are referred in ϕ' , at least one delegation identifier object from this domain be referred in ϕ' . This establishes the convention that each subject and holder term used for referring to parties in the formula is introduced through an identifier object in the identifier object introduction sub-formula of ϕ . This is not a limiting assumption as additional pseudonyms can be generated automatically as part of the process of computing a formula fulfilling a policy expressed in the fragment of $\mathcal{L}^{\mathbf{r}/\mathbf{g}}$ of Sec. 4.6.

For each identifier object p_i , we additionally require that its corresponding object specification formula ϕ_i only be part of ϕ_1, \dots, ϕ_k if p_i has been established with the intended recipient \mathbf{B} of ϕ' . Whether this holds can be determined through metadata, because the recipient is not part of the logic-based formal data model.

As another constraint, a formula ϕ' always refers to the holder and subject attributes through constant terms and does not relate those attributes otherwise, that is, through equality relations without revealing them. This unifies the processing of those formulae when run-time generating the underlying cryptographic credential protocols, and there is no need to hide those terms from the other party as they represent identifiers revealed as part of the establishment of the identifier objects they originate from.

The data statement ϕ' uses, when being constructed by \mathbf{A} , only the terms referred to in the object specification formulae ϕ_1, \dots, ϕ_k represented locally in \mathbf{A} 's repository that specify the objects referred to in ϕ' . Construction of ϕ' from the formulae $\phi_i, 1 \leq i \leq k$ requires that parts of the formulae ϕ_i be taken over into ϕ' , or parts of ϕ' be logically inferred from (parts of) the formulae ϕ_i .

Party \mathbf{A} releases a formula ϕ constructed from ϕ' to party \mathbf{B} , where the terms used for specifying the *holder* and *subject* may differ between ϕ' and ϕ . That is, the formulae ϕ_i held by party \mathbf{A} as part of its identity and identifier relationships and the formula ϕ to be generated from those can use different terms for referring to certain attributes or objects. This holds for both the identifier objects as well as the identities referred to in the formula. The transformation from a formula ϕ' to a formula ϕ that can be revealed comprises two steps, (1) a transformation related to the holder and subject terms, and (2) a replacement of terms referring to identities, opaque identities, and conditionally released identities.

4.8.2.1 Holder and Subject Term Transformation

The replacement of holder and subject terms in ϕ' is a first transformation step executed on ϕ' for obtaining ϕ . The basic idea is that holder and subject terms used locally by \mathbf{A} are replaced with terms associated with identifier objects as computed during their establishment based on the (cryptographic) protocols. In addition to such replacements, certain logical equality relations on renamed terms need to be introduced to make the result ϕ logically equivalent to ϕ' .

The need for this replacement of terms arises from the approach of using, for all identifier objects, their associated subject and holder terms as defined through the underlying cryptographic object in data statements because correctness of those can be verified through the cryptographic protocol for releasing data. See Sec. 5.4.6 for details on how a formula ϕ is derived from (a subset of) the party's repository formulae ϕ_i .

We next formalize the renaming of the terms for a formula ϕ' in a general setting of the formula referring to multiple identifier objects with subjects being \mathbf{A} or additionally multiple delegaters, where the objects may be from different registration domains or delegation registration domains. Note that we use, w.l.o.g., simplified indexing of entities compared to the discussion above on their establishment.

This term transformation is performed before computation of the cryptographic protocol to ensure that both prover and verifier use equivalent input formulae based on which to execute their respective cryptographic protocol for computing the to-be-generated cryptographic objects and proving the formula correct and verifying the created proof against the formula.

The formulae $\widehat{\Phi} = \{\phi_1, \dots, \phi_k\}$ specify their corresponding identifier objects and identities and are part of corresponding identifier and identity relationships. The terms used for referring to their holders and subjects are those as discussed in detail further above. Let, without loss of generality, $\Phi = \{\phi_1, \dots, \phi_\beta\}$ be object specification formulae specifying the corresponding identifier objects $\mathcal{Q}' = \{p_1, \dots, p_\beta\}$. Let \mathfrak{E} be an equivalence relation on Φ such that for (ϕ_i, ϕ_j) , with $i \neq j$, it holds that $(\phi_i, \phi_j) \in \mathfrak{E}$ if and only if ϕ_i contains the predicate $p_i.\text{holder} = t$ and ϕ_j contains the predicate $p_j.\text{holder} = t$ for a constant term t . Let, w.l.o.g. due to simplified indexing, the partition on Φ induced by the relation \mathfrak{E} comprise the equivalence classes Φ_1, \dots, Φ_ℓ with $\Phi_1 = \{\phi_1, \dots, \phi_{\xi_1}\}$, where $1 \leq \xi_1 \leq \beta$, and, if $\ell > 1$, for $1 < \gamma \leq \ell$, $\Phi_\gamma = \{\phi_{\xi_{\gamma-1}+1}, \dots, \phi_{\xi_\gamma}\}$, where $\xi_{\gamma-1} < \xi_\gamma$ and $1 < \xi_\gamma \leq \beta$.

An equivalence class Φ_γ corresponds to the identifier objects with their holder being represented through the same term in the formula ϕ' , that is, identifier objects part of the same registration domain.

For each equivalence class Φ_γ for $1 \leq \gamma \leq \ell$, let \mathfrak{E}'_γ be an equivalence relation on Φ_γ such that for all $(\phi_i, \phi_j) \in \Phi_\gamma$, with $i \neq j$, $(\phi_i, \phi_j) \in \mathfrak{E}'_\gamma$ if and only if ϕ_i contains the predicate $p_i.\text{subject} = t$ and ϕ_j the predicate $p_j.\text{subject} = t$ for a constant term t , inducing, w.l.o.g. due to using an indexing scheme renaming the indexes to include the equivalence class identifier γ and using simplified indexing within the class analogous to above, a partition on Φ_γ with the equivalence classes $\Phi_{\gamma,1}, \dots, \Phi_{\gamma,\ell'_\gamma}$ as follows: $\Phi_{\gamma,1} = \{\phi_{\gamma,\xi_{\gamma-1}+1}, \dots, \phi_{\gamma,\hat{\xi}_1}\}$ with $\xi_{\gamma-1} + 1 \leq \hat{\xi}_1 \leq \xi_\gamma$ and $\xi_0 = 0$ and, if $\ell'_\gamma > 1$, $\Phi_{\gamma,\eta_\gamma} = \{\phi_{\gamma,\hat{\xi}_{\eta_\gamma-1}+1}, \dots, \phi_{\gamma,\hat{\xi}_{\eta_\gamma}}\}$ with $\hat{\xi}_{\eta_\gamma-1} < \hat{\xi}_{\eta_\gamma}$ for $1 < \eta_\gamma \leq \ell'_\gamma$.

Each equivalence class $\Phi_{\gamma,\eta_\gamma}$ corresponds to identifier objects specified through formulae ϕ_α having the same party as their subject, referred-to through the same term t_α^s . Let t_α^h be the holder term specified for the holder of p_α in ϕ_α and t_α^s the subject term.

Let, w.l.o.g., $\Phi_{\gamma,1}$ be the equivalence class of the partition induced by \mathfrak{E}'_γ where each identifier object has equal holder and subject terms, that is, both are referring to party **A**. The other classes $\Phi_{\gamma,\eta_\gamma}$ for $1 < \eta_\gamma \leq \ell'_\gamma$ have formulae specifying identifier objects with delegaters as subjects as elements. The renaming proceeds as follows, with first choosing the identifier object corresponding to one of the formulae of $\Phi_{\gamma,1}$, through following a well-defined procedure, e.g., choosing the first one being referred-to in ϕ' . Let, w.l.o.g., this identifier object be denoted as $p_{\gamma,1,1}$ and its holder and subject be $t_{\gamma,1,1}^h$.

Next, we explain the required steps to be performed on ϕ' for obtaining ϕ . Replace all occurrences of $t_{\gamma,1,1}^h$ in ϕ' with the term $t_{\gamma,1,1}^{h'} := \varrho_{\gamma,1,1}^{\text{idf}}(t_{\gamma,1,1}^h)$ resulting from applying the renaming function $\varrho_{\gamma,1,1}^{\text{idf}}$ for $p_{\gamma,1,1}$ on $t_{\gamma,1,1}^h$, except for in logical equality predicates expressed on the subject or holder attribute of other identifier

objects than $\mathbf{p}_{\gamma,1,1}$. For each of the other identifier objects $\mathbf{p}_{\gamma,1,\delta}$ of $\Phi_{\gamma,1}$, replace its holder and subject in equality relations with the term obtained from evaluating the renaming function $\varrho_{\gamma,1,\delta}^{\text{idf}}$ of $\mathbf{p}_{\gamma,1,\delta}$ on the respective term resulting in $\mathbf{t}_{\gamma,1,\delta}^h$ and introduce the object equality relation $\mathbf{t}_{\gamma,1,1}^h = \mathbf{t}_{\gamma,1,\delta}^h$. The equality relation ensures that the party equality semantics expressed in ϕ' through using the same term remains specified in ϕ after the renaming.

For each equivalence class $\Phi_{\gamma,\eta_\gamma}$ other than $\Phi_{\gamma,1}$, perform the following processing: Choose, through a well-defined procedure, one of the identifier objects of $\Phi_{\gamma,\eta_\gamma}$, and let, w.l.o.g., this be $\mathbf{p}_{\gamma,\eta_\gamma,1}$. Replace all instances of $\mathbf{t}_{\gamma,\eta_\gamma,1}^s$ in ϕ' with the term $\mathbf{t}_{\gamma,\eta_\gamma,1}^s := \varrho_{\gamma,\eta_\gamma,1}^{\text{idf}}(\mathbf{t}_{\gamma,\eta_\gamma,1}^s)$ obtained through evaluating the renaming function corresponding to $\mathbf{p}_{\gamma,\eta_\gamma,1}$ on the term, except for in logical equality relations expressed on the subject or holder attribute of other identifier objects than $\mathbf{p}_{\gamma,\eta_\gamma,1}$. Replace $\mathbf{t}_{\gamma,\eta_\gamma,1}^h$ with the corresponding term given by the renaming function. Replace, for each identifier object $\mathbf{p}_{\gamma,\eta_\gamma,\delta}$ of $\Phi_{\gamma,\eta_\gamma}$ other than $\mathbf{p}_{\gamma,\eta_\gamma,1}$, its holder term $\mathbf{t}_{\gamma,\eta_\gamma,\delta}^h$ with the term $\mathbf{t}_{\gamma,\eta_\gamma,\delta}^h$ resulting from the application of renaming function of $\mathbf{p}_{\gamma,\eta_\gamma,\delta}$ and the subject $\mathbf{t}_{\gamma,\eta_\gamma,\delta}^s$ with the term $\mathbf{t}_{\gamma,\eta_\gamma,\delta}^s$ obtained through application of the renaming function. Next, add the logical equality $\mathbf{t}_{\gamma,\eta_\gamma,1}^h = \mathbf{t}_{\gamma,\eta_\gamma,\delta}^h$ for the holder and logical equality relation $\mathbf{t}_{\gamma,\eta_\gamma,1}^s = \mathbf{t}_{\gamma,\eta_\gamma,\delta}^s$ for the subject to retain the party equality semantics of ϕ' in ϕ . Finally, the equality relation $\mathbf{t}_{\gamma,\eta_\gamma,1}^h = \mathbf{t}_{\gamma,1,1}^h$ is introduced for linking the terms corresponding to different equivalence classes.

After each equivalence class $\Phi_{\gamma,\eta_\gamma}$ has been processed following the explained approach, the obtained formula ϕ is logically equivalent to ϕ' . What we have achieved is that term names are used in compliance with the constraint expressed earlier of being cryptographically verifiable based on the underlying cryptographic objects, that is, as defined through the renaming function of the corresponding identifier object. This is a strong property we achieve, while at the same time having identifiers locally that allow parties to process their portfolio formulae.

Particularly, each logical equality relation expressed on the holder or subject terms $\mathbf{t}_{\gamma,\cdot,\cdot}^h$ and $\mathbf{t}_{\gamma',\cdot,\cdot}^h$ or $\mathbf{t}_{\gamma,1,\cdot}^s$ and $\mathbf{t}_{\gamma',1,\cdot}^s$ of identifier objects of different equivalence classes for bridging registration domains is properly handled by the term renaming and expresses the domain bridging using the terms of the obtained formula.

This concludes the formal specification of the term renaming for the terms referring to the subject and holder of objects. In Sec. 5.4.2, we present the term renaming for a less generic common case for pseudonym and credential protocols in detail.

4.8.2.2 Object Term Replacement

Similarly, the terms used to refer to the identities themselves may differ between local representation and use. This is crucial for the unlinkability properties and explained in detail in Chapter 5. To summarize, each identity term is substituted with a term derived from the cryptographic protocol for proving holdership of the underlying cryptographic object. That is, this renaming step can only be executed

once (part of) the cryptographic protocol has been computed by **A**.

Similarly, terms referring to opaque and conditionally released identities generated in the protocol are replaced with terms derived from the respective generated cryptographic objects. This is required in order to replace the temporary terms having been introduced for referring to those objects before their underlying cryptographic objects have been generated.

4.8.2.3 Discussion

A formula ϕ to be released is obtained from **A**'s internal representation ϕ' thereof, as discussed, through a renaming of the terms of the holder and subject attributes as well as the identities and conditionally released and opaque identities. This is done as part of the protocol for releasing data. The holder and subject terms used are those derived from the cryptographic object corresponding to the identifier object. For private certificate protocols, the terms for identities are derived from cryptographic values generated in the course of the protocol. The correctness of terms is verified by the recipient of the data statement through the cryptographic material.

Following the above, any use of a *holder* or *subject* attribute of an identity in a formula released to a recipient party is constrained such that only terms must be used for the attributes that are used for the *holder* or *subject* attribute of identifier objects established with the party the formula is intended to be released to. This reflects the natural property of an identifier object representing a shared identifier (pseudonym) for a party used by a party in communication with data recipients.

The case of objects from more than one registration domain being used for making statements about a party in a formula ϕ' results in a number of different terms used to refer to this party in holder and subject terms of the formula that equals the number of registration domains. After the renaming transformation, the resulting formula ϕ will refer to at least those identifiers and will have additional object equalities added for equating pairs of terms relating to identifier objects of the same registration domain.

If the renaming were not done following the cryptographic protocol, the resulting formula might provide possibilities for an attacker to violate the semantics in a subtle way. For example, for two bank accounts with different currencies of the same bank, an attacker could, in case of random terms for identities being permitted, use an improper identity name in a released data statement to a data recipient. This could result in a “mixing” of the two identities in an instance of reasoning over the formula and a previously released formula, thereby achieving that the reasoning party would derive a statement over the mixed identity comprising, e.g., the account balance from one and the currency from the other identity. This can be seen as being related to a general variant of the identity mixing problem discussed in related work [CMN⁺10].

4.8.2.4 Functions of Parties

Subject and holder terms are used in a formula to be released to refer to parties the formula makes statements about. The terms are introduced in the identifier object introduction sub-formula related to the primary identifier objects of the formula. At least one identifier object referring to the party releasing the data statement needs to be present. In a delegation relationship, the different parties, namely the delegatee and delegater, or multiple delegaters, are referred to through identifier objects. Their function of being holder or subject of the identifier objects and identities being referred to is specified through equality predicates. This allows for making statements about all parties in a formula and making clear what their roles are in a data statement. The data request language defined in Sec. 4.6 earlier in this chapter allows for expressing requirements related to both identifier objects and identities relating to the delegater and delegatee of a delegation relationship.

The standard case for the use of a credential system is that both holder and subject of an identity refer to the party holding the identity and using the identity for making statements over its attributes, that is, about itself. In this case, the interpretation of the subject term coincides with the interpretation of the holder term of the identity. Another important case is when an identity is delegated, that is, when its subject attribute is a term referring to a delegater that has delegated authority over the identity to the party. The holder in this case is again the party to whom the identity has been delegated. A third case is when the subject is a third party about which a statement is to be made without a delegation being in place, e.g., when a service provider releases data about a customer to another service provider. This case is not relevant in the context of credential protocols, though, is important for the overall identity management system.

The subject of a delegated identity in statements made by the delegatee can take on different subject terms in different uses of the delegated identity, much like it can be done for the party's own identities in a standard interaction without delegation. That is, multiple delegation identifier objects can be created by the delegatee with the delegater as subject and a delegated identity be bound to those and attribute statements of the delegated identity be revealed in interactions with data recipients. The underlying delegation secret key can be used much like a party's own secret key, only being restricted to at most all delegation identities with the same subject and by the same certifier.

In all the cases, the subject attribute is used for binding identities or identifier objects to the same subject, that is, expressing that the subject attributes refer to the same party. Analogously, the holder attribute can be used in all cases for binding holdership of identities to a single party, that is, the party that is making a statement and proof using credential protocols over multiple identities backed by certificates. For each of the above cases, identifier objects for the holder and subject are referred to in a formula to be released, for the non-delegation case a single such object can be sufficient.

Repeatedly using the same identifier objects in data statements towards a data recipient establishes the linking semantics of reusing pseudonyms and can augment

the pseudonyms with further data that are associated with them. This applies to both pseudonyms with the party releasing the data statement as well as such with delegaters as subject. The concept of pseudonyms as “false names” of parties, that is, semanticless identifiers, is important for the related discussions.

4.8.3 Conventional Certificates

The generic discussions above capture the semantics of term renaming relevant for cryptographic credential protocols as underlying technology for revealing certified data. Certain protocols based on online identity providers who vouch for statements on behalf of the subjects can benefit from the presented ideas alike.

Conventional certificates based on signature schemes such as RSA, DSA, ECDSA, or similar schemes and the application of an algebraic-structure-destroying hash function such as a SHA variant on the message to be signed expose differences in terms of how the data model is applied. We sketch how this is done next—the details are not given, though, the basic ideas should suffice for an understanding following the above discussions.

A signature key pair created by a party **A** corresponds to the creation of a new registration identifier object. From the public key, the *holder* and *subject* terms as well as a *subjectId* term can be computed. The registration identifier object induces a registration domain and is specified through an object specification formula making statements about the *holder*, *subject*, and *subjectId* through the computed terms. The public key gets disseminated to possibly all parties in the system, all of which refer to it using the same term related to the corresponding identifier object, as well as the same terms for its holder and subject. The term for the *subjectId* is the same one for every party by its definition.

A certificate issued by a certifier on such public key using a traditional signature scheme has a corresponding registration identity relationship and the certifier is a registration certifier. The terms used for referring to the holder and subject of the certificate are those created in the context of the identifier object the public key corresponds to. A certificate obtained by **A** corresponds to a new identity of **A** that is specified through an object specification formula. Much like for any identity, the specification formula and certificate are stored as part of an identity relationship.

Like the public key, the certificate gets disseminated to possibly all parties in the system, each of which refers to it with the same term and uses the same terms for its holder and subject **A**. Each party obtaining the certificate stores it as an entry in its profile data, with the corresponding object specification formula specifying the certified attributes and their values. If the certified public key is the public credential verification key corresponding to an identity relationship of such other party, the term used to refer to the certifier of the identity of such identity relationship is that used for the identity relationship for certifying the public key. This model supports also PKI certificate renewal, where holder and subject terms remain unchanged for a new certificate issued on a given key.

Aspects such as generalizing this as for credential protocols as shown further above is possible in our general approach, though, traditional certificate technology

does not support this well and excessive information is released in each transaction. Such extensions may comprise delegation or having multiple identities in the same registration domain.

For each identity relationship a party establishes based on any kind of certificate, the party always establishes an entry in its profile data repository for holding the public key certificate for the public key corresponding to the certificate of the identity relationship, including the key, and the corresponding specification formula. For the credential protocols of our interest, those profile data entries are crucial for the process of finding a fulfilling formula based on a party's identifier and identity relationships for a data request as discussed in Sec. 5.4.6: The specification formulae of those profile entries comprise the information about the certifiers of the identity relationships the party holds and thereby are required for finding formulae that fulfill a data request of another party.

The above discussions apply to signature key pairs using the RSA, DSA, ECDSA, or similar schemes and corresponding certificates as well as Camenisch–Lysyanskaya signature key pairs [CL02b, CL04] used for our credential protocols and corresponding private certificates. The further models key pairs and certificates used for establishing secured communication channels, for realizing PKI hierarchies, or for parties to assert attribute statements to other parties based on conventional attribute certificates analogous to credential protocols, only with substantially less privacy. The latter models the certification of credential signing keys of certifiers used for issuing credentials. This gives rise to an architecture applicable independent of the kind of party or the technology being used for protecting the integrity of attributes, including standard PKI technology and extensions to PKI approaches for public key certificates on credential system public keys.

The case of a party signing the public key of a Camenisch–Lysyanskaya signature key pair with a Fiat-Shamir-type signature based on a data statement it proves using credential protocols does not fit this key certification model perfectly. The difference is that there is no single certifying party of the key that would mandate the establishment of an identity relationship. Rather, one or more identifier and identity relationships are used for creating the signature over the key. The semantics of the signature is still that the certifiers of the used identity relationships endorse the data statement made in the context of the Fiat-Shamir signature. The Camenisch–Lysyanskaya private signature key in this case is not cryptographically bound to the party having signed it through a Fiat-Shamir signature. The best that can be achieved with our protocols is that a cryptographic proof of knowledge of the private signature key corresponding to the certified public key be made and included in the resulting public key certificate. Our approach of realizing pseudonymous certifiers is also captured in the secure channel model of Chapter 3 in that a certifier is known to other parties under an authenticated statement in a generic way.

The above application of the data model to traditional certificate technologies and Fiat-Shamir public key certificates shows very well how we can handle different kinds of certificates and keys of parties acting in different roles in a unified way through a party-symmetric architecture. Particularly, the different entities are all modeled using the same approach to data representation. The advantages of this

show in the data integration for automated reasoning and particularly the support of ontologies for computing how to fulfill a data request or checking whether a statement fulfills a request.

4.8.4 Discussion

Regarding technology, cryptographic pseudonyms as underlying cryptographic objects for identifier objects can be implemented through cryptographic means, e.g., through privacy-enhanced pseudonym and credential protocols such as that of Camenisch and Lysyanskaya [CL01]. Alternatively, the implementation can be based on trust in the party, e.g., a service provider releasing data about one of its customers to another service provider and making a claim about the subject identifier. In the further case of private certificate systems, derivation of terms and enforcement of the use of only correct subject terms is realized through cryptographic means to prevent attacks by dishonest users, while in the latter case trust in the service provider is required that it is using the correct terms.

Overall, the seemingly complex (re)naming of terms in formulae of the local data representation and formulae to be released serves the purpose of allowing for using the calculi based on our first-order-logic-based language for making derivations and, at the same time, having a well-defined semantics for this language.

Regarding the modeling of data, we note that the introduction of the existential quantifier for data statements would allow for using a different modeling approach where the party identifiers and identity names would be expressed using variables bound through existential quantifiers. That is, the use of term renaming for such entities could be substituted with variables quantified existentially. Overall, we found this approach, though conceptually clean, to be more complex in terms of the resulting language and reasoning over it. Thus, the renaming of terms as outlined above and also in Chapter 5 has been chosen for our system.

4.9 Calculus

We next give an overview of the logic we use for representing our data model and the deductive system for making inferences over formulae in the model. Regarding foundations and terminology related to those aspects, we build on and refer the reader to work in the domain [HR04, Gir90, Bus98, Pfe04]. The main goal of this section is to point out key techniques related to implementing a deduction system for our logic and related challenges. An in-depth discussion on an efficient deduction system for our logic, including the prevention of the effect of contradictions introduced by an attacker, is left to future work.

Our system and framework comprises multiple languages based on our data model for different purposes of representing data and interfacing with cryptographic protocols. Those languages follow the same concepts and syntax. The differences between the languages are that they are constrained according to their intended use, e.g., specifying the release of data using a concrete protocol, or specifying

the establishment of an identity relationship based on a private certificate. The statement-type languages, e.g., for expressing object specification formulae and data statements, have a single semantics as discussed in Sec. 4.11. Also, there is a single set of axioms defined to make derivations in the logic.

We chose to build our calculus and languages on *first-order logic* for its high expressivity, simplicity, and available (open source) tool support. We use an extension of a fragment of first-order logic, the main aspects of which are detailed next.

4.9.1 Properties of the Logic

Logic with equality. We use first-order logic with *equality*, that is, with the symbol $=$ expressing object equality. The logic symbol $=$ allows for expressing that two terms refer to the same object in an interpretation, even though the terms may not be equal. This implies not building on the *unique names assumption (UNA)* which states that different constants refer to different objects in every interpretation, that is, every object is referred to through a single, unique, name in the syntax. This consequently results in a logic which is not a Herbrand logic.

Equality and not using the unique names assumption is required for reaching the desired degree of expressivity in terms of privacy in our modeling approach as it allows for using different terms for referring to the same object, e.g., an identity or holder or subject term. Object equality can relate such different terms with an equality of the underlying object in an interpretation. Whenever there is a need for specifying sameness of the objects referred-to through two (different) terms t and s in a formula, an equation $t = s$ is introduced to indicate the equality.

Sorted logic. We employ a *typed logic*, also denoted as sorted logic or many-sorted logic, meaning that each term has a “built-in” type from a type system fixed with the logic. A typed first-order logic is a standard extension of pure first-order logic where the types are associated with all terms of the language. Any many-sorted first-order logic with a finite number of types can be reduced to single-sorted first-order logic.

In contrast to the language of Camenisch et al. [CMN⁺10] which uses sorts of the logic also for the (sub)typing of identities, we use an even more flexible approach to (sub)typing identities for gaining substantial advantages in the obtained expressivity of the language. Only formulae being correctly typed are valid and further used for processing by a party.

Open world assumption. Our logic builds on the open world assumption, that is, there are no defaults as in calculi based on the closed-world assumption. This fits well our purpose of modeling knowledge of aspects of the real world relevant to our system and the incomplete way of specifying knowledge through ontologies, where parties may only obtain a subset of the defined ontologies. A closed-world logic of defaults may lead to inappropriate conclusions being deducible in such setting.

Integer extensions. Our data model requires that extensions for supporting types with a total order and very large, or infinite, value domains be in place. The data types *int* and the different date types and predicates expressed over those exploiting the total order of those types require this, where the latter types additionally require support for the specific relations between dates of different granularities as discussed in Sec. 4.4.13. We will refer to the related extensions as *integer extensions* in this work.

4.9.2 Deductive System

A main reason for our choice of building our formalism on top of first-order logic are the *deduction systems* that are available for such logic. A deduction system is also referred to as deductive system, reasoning system, calculus, or deduction apparatus. A main concern of the science of logic is reasoning, that is making formal inferences in the logic. Because a deductive system can be applied to prove formulae correct under given assumptions, it is frequently also referred to as *proof system*. Henceforth, we denote the proof system of our calculus as \mathcal{L} .

A basic concept related to deduction is the *sequent*, a standard logic concept that expresses that a formula ψ , the *succedent*, can be deduced, or inferred, from a list of formulae $\Gamma = \phi_1, \dots, \phi_k$, the *antecedent*, through a given deductive system of the logic:

$$\phi_1, \dots, \phi_k \vdash \psi .$$

Antecedent and succedent are referred to as *cedents*. A sequent is fulfilled if the succedent can be formally inferred (deduced) from the antecedent, that is, assuming the formulae in the antecedent are true, the succedent is true. In other words, the antecedent expresses the *hypotheses* to hold, while the succedent models the *conclusion* that can be inferred. In this thesis we use sequents to explicate formal derivations to be done in the deduction system for our logic. We also allow for the notation of having sets of formulae in the antecedent, with the meaning that all formulae of a set are part of the list of formulae.

The succedent can, in general, comprise a list of formulae ψ_1, \dots, ψ_l instead of a single formula. For our treatment, the case of a single formulae forming the succedent is sufficient. A sequent as presented above is equivalent to the conjunction of formulae in the antecedent implying the disjunction of formulae in the succedent being a tautology:

$$\vdash \bigwedge_{i=1}^k \phi_i \rightarrow \bigvee_{i=1}^l \psi_i .$$

An *environment* \mathcal{E} , following standard terminology in logic [HR04], is a function from free variables of a formula to constants of corresponding sorts, that is, an assignment of free variables with values from the value domains of those variables. We require that all free variables in all formulae in all cedents be instantiated through an *environment* \mathcal{E} or a free variable have the meaning that any of its possible

assignments results in the same denotation, before invoking the deduction apparatus for the sequent.

Deduction and proofs. Deduction in a formal logic is done based on a proof system for the logic, through an application of rules and use of logical axioms. A plurality of deduction systems with the same deductive power, differentiated through their rules, axioms, and proof strategies, can be defined for a given logic. For first-order logic, various seminal calculi have been proposed. Most prominently, this comprises the Hilbert or Frege proof system, natural deduction, the sequent calculus, and the resolution proof system—the reader is referred to work in the logic domain [Bus98, Gir90, Wal11, HR04] for an overview of the calculi.

A deductive system for a language is defined independent of any interpretation of the language, that is, it is only about syntactic consequences following from the logic's proof theory. For first-order logic, completeness and soundness of the deduction systems mentioned below is a well-established result. Completeness means that every formula that is true in an interpretation can be inferred in the deductive system. Soundness means that every formula that can be inferred is true in an interpretation.

Proofs in a proof system are often denoted as a tree-like structure—rooted at the bottom—from the axioms assumed to hold at the top to the proof goal at the bottom, and each step of the deduction procedure corresponding to an inner node of the proof tree. Using such notation, a deduction can proceed in a bottom-up way from the goal to be proven to the axioms, or top down, starting from the axioms, or as a combination thereof. It can be observed that, unless the search space is substantially constrained during this selection process, a top-down approach will result in a search space of unmanageable size [Bus98] for general first-order logic.

Practical deductive systems. *Hilbert-style proof systems* are based on multiple axioms or axiom schemas and only one or two deduction rules and result in lengthy proofs. Although the system is simple and powerful in terms of its deductive capabilities, implementing computerized procedures for finding proofs in Hilbert-style proof systems has proven to be hard [Bus98].

Natural deduction has been proposed by Gentzen [Gen35] as an approach that comes as close as possible to human reasoning. Natural deduction calculi operate on a comprehensive set of deduction rules while not requiring axioms for basic first-order logic. The resulting calculus is extremely intuitive to handle by humans and comprises applying a rule to derive a new theorem valid under the given axioms in each deduction step, operating in a top-down manner. The main drawback of natural deduction is that it cannot be reasonably implemented as a computerized algorithm due to the huge search space and the difficulty of realizing proof strategies that reduce the search space [Bus98, HR04].

The *sequent calculus* by Gentzen [Gen35] is based on a comprehensive set of rules compared to Hilbert-style calculi. Each step in a proof in the sequent calculus is a sequent constructed through application of a single rule, where deduction operates

in a bottom-up manner, that is, starting from the proof goal. The sequent calculus has been introduced by Gentzen as an extension to natural deduction. It is arguably the most elegant and flexible system for writing proofs [Bus98]. The sequent calculus is closely related to natural deduction and can be obtained by first applying natural deduction introduction rules both bottom-up and elimination rules top-down in a derivation, resulting in a so-called intercalation calculus, and then reformulating the rules to obtain a bottom-up-only calculus [Pfe04]. The sequent calculus can be extended towards realizing practical automated deduction systems [Pfe04], though, is not practically applicable for automated deduction in its pure version [Bus98].

Today's most-widely used approach to deduction is *resolution*, resulting in a refutation-based deduction procedure. Resolution has been introduced by Robinson [Rob65] and Davis and Putnam [DP60], for propositional and first-order logic, respectively. This approach is essentially based on a proof through contradiction, that is, the negation of the formula to be proven is used in the process instead of the proof goal itself. In each step, the resolution rule is applied—the only derivation rule in the resolution calculus. Finding a contradiction proves the to-be-proven formula true. Resolution results in efficient proof search procedures as well as proofs the lengths of which are reasonable, thereby making it a practical scheme [Bus98].

A logic with equality requires, for not being a Herbrand logic, specific techniques for its deduction apparatus. Basically, equality can either be handled axiomatically through adding the equality axioms to the axioms of the calculus, which results in a substantial increase of the proof search space by not considering the inherent structure of equality in the reasoning [Pfe04], or through handling equality through the inference mechanism [Bus98] directly. *Paramodulation* [RW69, WR73] is an example for the latter approach in the context of resolution-based deduction systems and is employed in a variant in leading resolution-based theorem provers, such as the example provers mentioned further below.

Any deduction system needs to implement a *proof strategy*, that is, a selection on which rules to apply in a given step of the deduction—for practical problem instances many possibilities exist in each step, thus giving rise to a huge overall search space for a given instance of deduction. The proof strategy is a key determining factor for the efficiency of the resulting proof system and its machine implementability.

A large literature body exists on proof theory and efficient machine implementations of deductive systems. Our overview has only presented the basics of those systems, the reader is referred to Pfenning [Pfe04] or Buss [Bus98] for further details and references.

4.9.3 Axioms and Theories

The *logical axioms* \mathfrak{X} are the axioms that form, together with the deduction rules, the deductive system \mathfrak{L} for a logic. \mathfrak{X} comprises the logical axioms for the calculus to be implemented through a reasoning engine. For a logic with equality like ours, it also comprises the equality axioms in case equality is to be realized through axioms, which is not recommended. The logical axioms are implicit in the notation $\Gamma \vdash \psi$ of ψ being a consequence of the premises Γ . A theorem prover implements the calculus

based on logical axioms and deduction rules and is used on sequents of the form $\Gamma \vdash \psi$.

According to Mendelson [Men87], so-called *proper axioms*, or non-logical axioms, are specific to the theory being considered and outside of a generic deduction apparatus for first-order logic. A *theory* over a language is the closure of the theorems that can be derived under both logical and proper axioms and the rules of deduction [Men87]. The theorems of the resulting theory induced by the proper axioms, the logical axioms, and the deduction rules are true only under interpretations that are compliant with the structure induced by the axioms, while the theorems that can be derived through the deduction rules from only the logical axioms are true under any interpretation. We denote a theory based on the proper axioms \mathfrak{J} as $\mathfrak{T}^{\mathfrak{J}}$. The deduction system \mathfrak{L} is implicit in this notation and assumed to be fixed. Such theory is also referred to as *deductive theory*, as being defined through a deduction apparatus.

In applications of logic, proper axioms are used to model the structure of a specific problem domain, e.g., an arithmetic [Men87]. In this thesis, we use proper axioms for modeling certain structural aspects of our identity management system, e.g., the typing system for identity subtyping (Sec. 4.4.11), or for modeling concrete relations valid in a given system, e.g., expressing the certifier ontology a party relies on in its deductions (Sec. 4.10).

Using knowledge representation terminology, an axiom can be either a *fact*, that is, a predicate that holds, or a *rule* specifying how to derive predicates that hold based on premises assumed to be true. Also note that inference over a given theory in a formal language is a purely syntactical process and fully determined through the underlying formulae.

The theory of interest for making derivations in our system considers also the proper axioms \mathfrak{J} for modeling our domain. As noted, it is henceforth referred to as $\mathfrak{T}^{\mathfrak{J}}$ and we use the notation $\Gamma \vdash_{\mathfrak{J}} \psi$ to indicate that a derivation be made in this theory, using our standard calculus \mathfrak{L} .

Technically, deduction for $\Gamma \vdash_{\mathfrak{J}} \psi$ can be realized through a deduction for $\Gamma, \mathfrak{J} \vdash \psi$ using a theorem prover, that is, the proper axioms are expressed as premises of the derivation to be made.

Note that deductions on a sequent require specific handling of free variables. An environment \mathcal{E} , that is, a function from free variables can be given as additional argument to a sequent, resulting in the notation of (4.21). The sequent is evaluated with the free variables in the preimage of the function \mathcal{E} being substituted with the terms according to \mathcal{E} . A free variable which is not instantiated through an environment has the meaning that it can be instantiated with any of the values of its type.

$$\Gamma \vdash_{\mathfrak{J}}^{\mathcal{E}} \psi \tag{4.21}$$

With the proper axioms expressed as part of the premises, this gives

$$\Gamma, \mathfrak{J} \vdash^{\mathcal{E}} \psi .$$

4.9.4 Machine Implementation

Automated reasoning in a calculus is a powerful tool for considering formally modeled knowledge of certain aspects of a system in the formal data model. Automated reasoning concretely allows for deriving, based on inputs that are assumed to hold, e.g., data statements proven by another party, facts that hold in the logic. The reasoning is specified through deduction rules.

An algorithmic implementation of a deductive system to be executed by a computing apparatus is referred to as *theorem prover*. It allows for computing in an automated fashion whether a sequent $\Gamma \vdash \psi$ holds. For computing fulfillment of sequents in the context of our data representation language, multiple approaches can be followed: A prototypical implementation can utilize an existing theorem prover, or a special-purpose theorem prover can be constructed.

Multiple efficient theorem provers for full first-order logic with equality have been implemented and made available under flexible open source licenses, where prominent provers include, e.g., the *E Theorem Prover* [Sch, Sch02] and *SPASS* [SPA10, WDF⁺09]. Both of those support full first-order logic with equality, based on resolution and a variant of paramodulation. Another example for a theorem prover capable of first-order logic is *Prover 9* [McC10]. Those tools apply numerous specialized techniques for increasing the efficiency of their deduction procedures. It is left to future work to investigate the applicability of specific provers to our logic with equality and integer extensions, as well as their extensibility, for obtaining a prototype implementation of a concrete deductive system for our logic. Also, the application of the advanced concepts utilized for realizing efficient deduction procedures in the best theorem provers for a specialized implementation of a deduction engine for our logic and its specifics, e.g., the arithmetic extensions for integers and types, might be worthwhile.

For building a production system, the approach of constructing a special-purpose deduction system is preferable to using an extension of a generic theorem prover. Identity statements in our language have substantial structure, thus comprise only a tiny fragment of full first-order logic with the discussed extensions. A special-purpose implementation can exploit the known structure of our language in the form of heuristics and the proof search strategy, compared to provers for general first-order languages. This is expected to reduce the proof search space, and thus runtime, considerably. Regarding the deduction mechanism, both the approaches of building either on the ideas of the sequent calculus or resolution can lead to an efficient practical implementation when exploiting the language structure.

Finding satisfying formulae for fulfilling a data request—see Sec. 5.4.6—requires additional functionality than only proving whether a sequent holds. A fulfilling formula, derivable in the theory \mathfrak{T} used for reasoning, as well as an environment specifying the variable assignment for the free variables of the conclusion need to be computed in addition to executing the theorem proving procedure. Thereby, the conclusion ψ (conjecture to be proven) is interpreted as a question and the theorem prover computes, based on the portfolio formulae ϕ_1, \dots, ϕ_k , a formula ϕ from which the conjecture is derivable under an also-computed environment \mathcal{E} . The process of

finding an answer for a conjecture is, in the context of automated theorem proving, denoted as *answer extraction*, see, e.g., Sutcliffe et al. [SYT09]. The variables of the conjecture to be instantiated are outermost existentially quantified in their formalism, while they are free variables in ours, for which reason we need an additional environment for instantiating the variables.

Investigating how answer extraction integrates with a deduction procedure of our logic, that is, how it can be integrated into the proof search, is a further interesting piece of future work. Available theorem provers support answer extraction as part of the search process—similar ideas can be employed for realizing a custom automated deduction engine for our language. The structure of our language benefits a substantially reduced search space of possible assignments to be attempted in a proof search compared to general first-order languages. Integrating finding a potentially fulfilling formula with the proof search and not performing the steps of finding a potential answer and checking whether it is a correct answer sequentially is crucial for a practical system in order to leverage ontologies as part of the used theory for finding an answer and for reasons of efficiency.

The language extension of supporting operators, see Sec. 4.4.17 further complicates practical implementations by further deviating from standard integer extensions to first-order logic. Similarly, the language feature of date types of different granularities and the relations over them complicate the reasoning. Thus, a production system may forego those features for a substantially simpler and more efficient reasoning engine.

4.9.5 Contradictions

The approach of resolution is based on the principle of proof through contradiction. Thus, when employing a deduction apparatus using resolution, care must be taken that the axioms of the theory together with the formulae in the antecedent do not lead to a contradiction on their own, which is an issue introduced through the openness of our system. A contradiction would let one infer any statement as true under the assumption of the hypotheses, which can allow an attacker to illegitimately gain access to a resource by introducing specifically crafted statements or ontologies (axioms) leading to a contradictory theory under which resolution would be performed.

Using deduction techniques not based on refutation may suffer from the same problem of contradictions. For such approaches, it is left open to investigate the possibility of adapting the deduction system to detect contradictions occurring in a deduction process or removing the deduction rules based on contradictions. This is possible in calculi where contradictions are not inherent to the approach of reasoning. Such changes of a calculus may lead to a calculus which is not complete in terms of allowing for the syntactic derivation of all logically entailed formulae. Lacking the completeness property may be tolerable, though, for our application. In the worst case, this may lead to (rare) situations of a policy not being satisfied through data statements in their evaluation, although logic entailment holds. Always retaining the soundness property, though, is clearly crucial for the security of the system.

It is worth considering how we can prevent an attacker from introducing formulae into the hypotheses used for an instance of deduction by providing a sequence of maliciously crafted formulae to the party executing a deduction procedure. The relevance of this lies in the application of deduction for the computation of authorization decisions by a party based on formulae received from a requester. This requires consideration in the axioms used as part of the theory, as well as the formulae provided by the requester, acting as premises for an instance of deduction.

Axioms. A possibility of contradictions being introduced is through formulae forming a part of an axiom system used as a subset of the constituting axioms of a theory. This can be countered irrespective of the deduction system being used. We distinguish the cases of (1) the system-specific non-logical axioms, (2) ontologies from trusted ontology providers, and (3) ontologies from untrusted ontology providers. Note that the logical axioms are fixed and determined by the choice of deduction system.

Case (1): The system-specific axioms need to be designed carefully to not induce a contradiction in any theory based on legitimate formulae as axioms and reasoning thereover based on legitimate premises. An example is the set of rules related to the type system for identities of Sec. 4.4.11.2. Case (2): Similarly, ontologies from trusted ontology providers need to be carefully designed to avoid inducing a contradiction in conjunction with legitimate formulae. Additionally, the language governing the permissible ontology rules is further constrained than for Case (1) to not cause conflicts in conjunction with the system-specific axioms. An example for such ontologies are the certifier ontologies of Sec. 4.10.1. Case (3): Ontologies from untrusted ontology providers are heavily constrained in their expressivity such that they can be verified by the recipient to not be crafted in ways to be able to induce contradictions together with the axiom parts of the theories they are to be used in. An example for this are the rules related to each identity type and required for considering such identity type in the automated reasoning framework.

Premises. Regarding formulae to be released, we have taken care in our language design to avoid that an attacker may be able to induce a contradiction in the theory used for an instance of deduction. Multiple provisions in the language design for taking care of this are presented next, while we leave, as for the above, an exhaustive treatment of this subject to future work. (1) Through the association of attributes and their values with identities instead of parties directly, a party can have multiple different values for a single attribute in different identities. This makes sense in many circumstances, such as nicknames. (2) The naming of terms based on underlying cryptographic protocols, e.g., zero-knowledge proofs of knowledge for proving knowledge of the corresponding cryptographic object, ensures that an attacker cannot use the same term for two different identities. Combining (1) and (2) prevents an attacker from creating a contradiction through the attribute values of illegitimate identities. Even by obtaining maliciously issued attributes in an identity certified by a certifier controlled by the attacker, the attacker cannot induce a contradiction through a particularly chosen attribute value by expressing predicates over this

value.³⁴ (3) An identity referred to in a data statement needs to be referred to by its base type in the type hierarchy to avoid contradictions introduced through different references of the same identity, which can be accomplished through the cryptographic proof in our credential protocols.

We also leave it open for future work how one can verify a formula or set of formulae of the premises for contradiction freeness in the theory of the set of axioms that need to be considered, also considering the approaches from above, to guarantee that no contradiction is induced by an attacker.

4.10 Ontologies

Gruber [Gru93] has defined an *ontology* as an “explicit specification of a conceptualization,” which is the most widely used ontology definition in use today [GOS04]. Borst [Bor97] defines an ontology to be a “formal specification” and particularly a “shared conceptualization.” This aspect of ontologies being a *shared* conceptualization is crucial for our work.

Technically speaking, in terms of our data representation language, an ontology is a *set of formulae* used for modeling a part of the domain of identity management of interest, on the one hand in terms of its structural properties, and on the other hand in terms of concrete instances of parties and other entities. The formulae of an ontology are used by parties as *axioms* in the course of automated deduction, thereby resulting in a specific theory derivations are made in.

Ontologies can be provided by third parties or defined by a party itself, where ontologies of the further form are denoted as *third-party ontologies*. The strength of using ontologies in our system comes largely from using third-party ontologies, because this allows parties to outsource certain decisions and assessments to those third parties instead of every party making those decisions and performing the assessments on its own. We find the support for such outsourcing to be a crucial functionality of an open identity management system, because assuming every party to be required to acquire or have the extensive knowledge expressed in the ontologies they use on their own is unreasonable.

Ontologies are put to multiple uses in our system, with a main use case being the specification of expressive data requests through referring to abstractions defined through ontologies. Another important use case is realizing certain system features, e.g., the binding of identities to a registration identity, where the certifier-permitted registration identities are specified through an ontology issued by the certifier. Through those and further uses of ontologies, they are crucial for enabling *interoperability* in an open system.

Based on the discussions in Sec. 4.9 on derivations over theories using proper axioms, we can compute derivations based on the ontologies $\mathcal{O}_1, \dots, \mathcal{O}_k$ as follows,

³⁴Note that this relies on a random challenge being provided by the verifier in the cryptographic credential protocol, ensuring that cryptographic values from which object identifiers are derived cannot be under the control of the attacker.

with the formulae of the ontologies being considered as further premises besides Γ :

$$\Gamma, \mathcal{O}_1, \dots, \mathcal{O}_k \vdash \psi.$$

4.10.1 Types of Ontologies

We define the following types of ontologies for our architecture: system-defined axioms, concept specification ontologies, party assessment ontologies, subtype ontologies, registration binding ontologies, and dummy identity ontologies. This partitioning is motivated by a system perspective related to which parts of an overall ontology will be provided by each ontology provider as separate entities and by a functional partitioning.

From a technical perspective, that is, the formal language and the applicable deduction procedures, our system is open in terms of what can be specified through ontology formulae. Constraining the languages for the specific types of ontologies helps in avoiding unintended consequences when composing ontologies or obtaining ontologies from non-trusted parties.

System-defined axioms. The *system-defined axioms* \mathcal{O}^{ax} are the axioms defined as an integral part of the system in the context of this thesis or future extensions thereof. The permitted syntax for expressing the rules is not further constrained as those axioms are under full control of the system developers and thus can be considered as being trusted. The set of axioms is small and the axioms induce a structure on our domain of discourse. There is one single fixed set of those axioms applicable to all parties in a system and the axioms are not subject to the usual scheme for obtaining ontologies.

The type derivation rules for the type system for typing identities of Sec. 4.4.11.2 are an example for system-defined axioms. Those rules are an example of structural rules describing the domain of interest—subtype hierarchies in the example. They are shipped together with an implementation of the system and thus, by definition, are received from a trusted provider. There is no need for system-defined axioms to be obtained by a party from a third-party provider.

Concept specification ontologies. *Concept specification ontologies* define concepts and relate them to other concepts and attributes of identities, and thereby determine the vocabulary that data requests can be expressed over and impose a structure on the vocabulary. A concept is an (abstract) entity of interest in our identity management domain that is made explicit as an element of the vocabulary formulae can be expressed over through its definition in an ontology.

Examples for such concepts are the concept of the trust level of a party, the assurance level of attributes certified by a party, or a party being an OECD government certifier, or EFTA or EU government certifier. An abstract concept like in these examples is defined through a formula representing a rule specifying which premises need to hold on a given entity, e.g., party or identity, in order that a predicate expressing the concept to be specified on the entity holds. In case a concept

is not expressed as part of an inference rule, it can be specified outside of the set of formulae to establish it as part of the vocabulary.

Technically, the definition of ontology rules is based on standard ideas of first-order logic of expressing predicates assumed to hold as premises and specifying, through implication, predicates that can be derived as conclusion, where entities such as parties or identities are referred to through \forall -quantified variables.

Using concept specification ontologies is crucial for the definition of vocabularies and semantics thereof in an open identity management system and for realizing abstractions in the definition of data requests.

Example 4.25 (Assurance level) *The below example specifies a simple rule that lets one infer an assurance level of 7 of a party (certifier), referred to by variable C, based on it being an OECD government certifier, having a trust level of at least 7, and undergoing a periodic security audit. The concepts in the premises are assumed to be well-understood concepts defined in this or other ontologies. The variable types of C and s are defined as C :: pty, s :: int.*

$$\begin{aligned} &\forall C, s : \text{Property}(C, \text{"oecdGovCertifier"}) \wedge \text{Property}(C, \text{"securityLevel"}, s) \wedge \\ &\quad \text{Geq}(s, 7) \wedge \text{Property}(C, \text{"periodicSecurityAudit"}) \\ &\quad \rightarrow \text{Property}(C, \text{"assuranceLevel"}, 7) \end{aligned}$$

The idea behind the concept of an assurance level is that it represents an indication of assurance of attributes vouched for by the assessed party (certifier). It can be used for abstractly specifying eligible certifiers in a data request, thereby capturing a (potentially) large set of eligible certifiers for which such property can be inferred with a single simple statement expressed in the request. Assurance levels are an example for a practical concept for abstractly specifying the certification requirements of an attribute.

As can be seen from the example, the rule uses the \forall -quantifier over the entities to make the statements about and expresses the inference through implication. Multiple rules can be expressed, also as parts of different ontologies, inferring the same predicate based on different assumed facts.

Example 4.26 (Concepts) *Examples for high-level concepts useful for the specification of data requests in the context of authorization policies are "securitylevel", "physicalSecurityLevel", "orgSecurityLevel", or "trustLevel". Examples for concrete abstractions over parties in the system are "swissGovtEntity", "oecdGovtEntity", "eftaGovtEntity" or "euGovtEntity", which can be equally useful for the concise specification of data requests. A further property interesting in the context of attribute assurance is the strength of identity verification a certifier performs on the attributes it vouches for. The meaning of the above examples should be clear from their naming.*

In a practical system, the concepts defined through a concept specification ontology can evolve from international or industry standards or recommendations in

a domain. A widely agreed standard or recommendation can lead to a generally accepted ontology specifying the concepts contained therein, thus avoiding fragmentation of the vocabulary and semantics thereof in the case of a large number of ontologies. Ideally, a single or a few such ontologies being internationally agreed (and standardized) would benefit interoperability of parties greatly through an agreed vocabulary and semantics as a basis for interoperability and common understanding. In this thesis, we remain with the examples above and do not give a complete concept specification ontology.

When making inferences, a party uses n_{cs} concept specification ontologies $\mathcal{O}_1^{cs}, \dots, \mathcal{O}_{n_{cs}}^{cs}$ as part of the overall ontology employed for its reasoning, resulting in \mathcal{O}^{cs} defined as follows:

$$\mathcal{O}^{cs} := \bigcup_{i=1}^{n_{cs}} \mathcal{O}_i^{cs}.$$

Party assessment ontologies. Ontologies formally specifying assessments of parties are crucial for realizing the outsourcing of trust decisions through the assessment of parties by so-called trust providers. Such outsourcing of trust facilitates the openness and scalability of the identity exchange system. An assessment of a party comprises a third party assessing the party with respect to certain metrics, e.g., metrics corresponding to concepts defined in concept specification ontologies, and formalizing the assessment results of the party through a *party assessment ontology* made available to other parties. Trivial examples of metrics of an assessment are attributes of the legal identity of the party, e.g., as available from PKI certificates related to the party or other sources.

We currently define the *certifier ontology* as a main kind of ontology in this category. Such an ontology makes statements about certifiers in the system and particularly associates properties with the certifiers. Another type of ontology in the category of assessment ontologies is the *conditional recipient ontology* which comprises assessments of conditional release trustees. Both those ontologies, and possible future further similar kinds of ontologies for other classes of parties, are generically referred to as *party assessment ontologies* and can help simplify authoring of data requests and thus authorization policies and substantially increase their expressivity through exploiting the knowledge modeled in the ontologies.

The assessment result of a party is formalized through a set of predicates or formulae expressed about the party. The following shows how a single property value u for property p is associated with a party s through a property predicate:

$$\text{Property}(s, p, u) .$$

Using a trinary predicate $\text{Property}_{\text{pty} \times \text{str} \times \beta}$ allows for associating a property value $u :: \beta$ of property p with subject s . The subject is represented through a constant term s under which the party is known to others, usually its public pseudonym.³⁵ The

³⁵Although it is possible from a data model and system perspective to refer to pseudonymous parties in party assessment ontologies, this is at the current time only a niche use case.

subject term is one which is used for certifier identifier objects and identities, that is, the one uniquely derivable from PKI certificates related to the party. Following this approach integrates the formulae referring to the terms with our approach of data modeling and particularly making automated deductions.

Following the earlier discussions of Sec. 4.4.14 on modeling property predicates, a k -ary predicate can be used for expressing $(k - 2)$ -tuples or $(k - 1)$ -tuples of values as properties of a party, one of the arguments being used for representing the subject and, depending on the approach, another argument for the property. Property predicates can be expressed both in the assumption and conclusion part of implication-based ontology formulae.

Attributes of a party, e.g., as contained in a PKI certificate, are expressed through a formula of attribute predicates as for any kind of identity, expressed over an identity corresponding to the PKI certificate. They use the same (public) subject identifier of the party as in the property predicates associated with the party. Additionally, a formula expressed over an identifier object based on the public key can be expressed to specify the corresponding *subject* and *subjectId* terms. This relates to how our data model is used for modeling conventional PKI technology as discussed earlier in Sec. 4.8. Even without an assessment, having those formulae available for reasoning about the certifiers as part of the ontology facilitates openness of a system.

The following example predicate associates the property “trustLevel” with value 8 with the party referred to by the constant *s*:

Property(*s*, “trustLevel”, 8) .

Properties that can be expressed in an assessment ontology can be anything ranging from low-level properties associated with the party to aggregate properties such as a trustworthiness level assigned to the party following a well-defined assessment and evaluation procedure based on information obtained through an audit of the party. Examples for assessed properties of parties may include, but are not limited to, the ones presented in the context of the concept specification ontologies.

Example 4.27 (Party specification formula) *The following property predicate expresses that the party referred to through subject term *s* is an OECD government certifier:*

Property(*s*, “oecdGovCertifier”) .

For such approach of party assessment and related trust outsourcing to work, we need to assume existence of parties in the system who act as ontology providers issuing assessment ontologies to other parties and who are trusted by the other parties for issuing the ontologies, that is, to make statements they have verified diligently, e.g., through the assessment of parties through audits or other suitable means.

Considering the information that assessment ontologies can provide, those ontologies are a means for outsourcing trust decisions that are otherwise hard or practically infeasible to make in an open, world-wide system, particularly for small

organizations or users. Reasons for this are that a view of the system with sufficient coverage and possibly access to parties for their assessment is required, which may be a very strong assumption or require a substantial investment by a party.

Defining and agreeing on ontologies that assign properties to parties, e.g., certifiers, through attribute statements, is a huge effort on its own. Thus, reusing the knowledge captured in such assessment ontologies and thus amortizing the effort over a large set of parties is a facilitator for an open identity exchange system as the one we define.

A main idea behind the use of party assessment ontologies is that in an open data exchange system the idea of specifying parties through attributes instead of (unique) identities as done for users in privacy-preserving user-centric identity management is carried further to the specification of certifiers. The reason is that for the assurance of attribute statements, in the general case, not the identity of the attributes' certifier is relevant, but so are its properties, e.g., as assessed through independent organizations.

A party uses the party assessment ontology \mathcal{O}^{tPy} constructed as follows from n_{tPy} such ontologies $\mathcal{O}_1^{\text{tPy}}, \dots, \mathcal{O}_{n_{\text{tPy}}}^{\text{tPy}}$ through set union:

$$\mathcal{O}^{\text{tPy}} := \bigcup_{i=1}^{n_{\text{tPy}}} \mathcal{O}_i^{\text{tPy}} .$$

Subtype ontologies. A subtype ontology \mathcal{O}^{tp} defines a type hierarchy over a subset of the identity types in the system. A formula of the following form comprising a single predicate expresses that type \mathbf{t}' is a subtype of type \mathbf{t} :

$$\text{Subtype}_{\text{str} \times \text{str}}(\mathbf{t}', \mathbf{t}) .$$

Using the subtype rules of Sec. 4.4.11.2, which are part of the system-defined axioms, and predicates expressed over type attributes in the object specification formulae of identities and data statements based on those allows for inferring Type predicates that hold with respect to a subtype ontology \mathcal{O}^{tp} for a given identity.

Any identity type that is not referred to in a subtype ontology a party uses in its deductions forms its own trivial hierarchy of height zero comprising only itself. The type id is always assigned to any identity through the logic's type system.

A subtype hierarchy specifies the subtype relations from the view of the provider of such ontology for a subset of the identity types in the system. In an open system, multiple providers will issue subtype hierarchies. This raises the problem of composition of the ontologies. Composing two subtype ontologies can, unless they are disjoint in terms of the identity types they talk about, lead to multiple supertypes being assigned simultaneously to a given identity in the reasoning procedure. This is related to multiple inheritance, though, based on different merged subtype hierarchies. That is, an identity can be the subtype of different supertypes of different type hierarchies the party relies on. We leave it to future work to explore the detailed implications and potential unintended consequences of this for applications of reasoning such as policy matching or computing authorization decisions.

Subtype hierarchies that are disjoint in terms of the identity types they refer to do not lead to composition problems and can thus be composed without unintended consequences. This situation can be achieved from having domain-specific ontologies, where for each domain one dominant ontology gets adopted in the system.

A subtype ontology comprises a set of subtype formulae for representing the hierarchy. A party may use multiple subtype hierarchies $\mathcal{O}_1^{\text{tp}}, \dots, \mathcal{O}_{n_{\text{tp}}}^{\text{tp}}$ for its logic inferences, resulting in a compound subtype ontology \mathcal{O}^{tp} :

$$\mathcal{O}^{\text{tp}} := \bigcup_{i=1}^{n_{\text{tp}}} \mathcal{O}_i^{\text{tp}}.$$

Registration binding ontologies. As discussed in Sec. 3.9, an identity can be established based on a registration identity, that is, the first identity obtained for a registration domain. When using credential systems, this ensures that the same private key of the party is used for issuing both the registration credential and the one to be issued based on it. Liability and trust issues are non-technical drivers that will, in practice, require that a certifier needs to be able to specify which registration identities (private certificates) it accepts for establishing its own identities (private certificates) based on those. This can be achieved through *registration binding ontologies* as explained next.

For credential systems, a registration binding ontology specifies, which registration certificates are permissible for binding to-be-issued certificates of a certifier to, using the same secret key. One such ontology specifies the permissible registration certificates for identities of one type one specific certifier issues. A party obtains such ontology from each issuer for each certificate type for all certificates it holds. Because in an open system any party can act as a certifier, certifiers are in general not trusted for providing ontologies. Thus, it must be ensured that the ontology be verified to not introduce unintended logical consequences into a deduction process using the ontology, or be generated through the party using it from a constrained formal representation of the ontology. Either of those approaches is equivalent in terms of security in that the ontology provider (certifier) cannot inject a rogue ontology into the reasoning process of the party.

For each permitted type \mathbf{t}_R of registration identity and certifier c_R for an identity of type \mathbf{t} with certifier c , a predicate is added to the ontology as follows, where \mathbf{t}_R and \mathbf{t} are constant terms of type str and c and c_R constant terms of type pty :

$$\text{PermittedRegIdType}(\mathbf{t}, c, \mathbf{t}_R, c_R).$$

The set of all such formulae for one pair (\mathbf{t}, c) comprises the registration binding ontology for the identity type \mathbf{t} , certified by the party referred to through subject term c .

Rule (4.22) is specified as part of the system-defined axioms \mathcal{O}^{ax} and models the structural relationships between identities in the system as induced by the relation $\text{PermittedRegIdType}$. The rule expresses the derivation of a predicate $\text{RegBind}_{\text{id} \times \text{id}}$

in case two identities are in an according relation as specified in the rule in (4.22).

$$\begin{aligned}
& \forall C, C_R, t, c, t_R, c_R : \text{Eq}(C_R.type, t_R) \wedge C_R.certifier = c_R \wedge \\
& \quad \text{Eq}(C.type, t) \wedge C.certifier = c \wedge \\
& \quad C.holder = C_R.holder \wedge \text{PermittedRegIdType}(t, c, t_R, c_R) \\
& \quad \rightarrow \text{RegBind}(C, C_R)
\end{aligned} \tag{4.22}$$

A data request can express the predicate $\text{RegBind}(C, C_R)$ on an identity variable C and thereby require that holdership of the single registration identity C_R with respect to which C has been established be required to be proven as well. A data request can express further predicates over C_R as needed for the policy at hand.

Due to the constrained nature of the identity binding ontologies and the formal verification or generation thereof through the party utilizing the ontology, composing such ontologies with each other or any of the other ontologies does not pose a problem.

A party should use registration binding ontologies, if available, for all identity types for which it has dummy identity ontologies to not unnecessarily constrain the automated deduction system when computing fulfilling statements for data requests. As discussed for dummy identities below, providers of certifier ontologies are, from a system perspective, in an excellent position to also vouch for registration binding ontologies for the referred-to identity types.

A party may use up to one registration binding ontology per active identity relationship in its repository and per dummy identity. Composing those ontologies $\mathcal{O}_1^{\text{reg}}, \dots, \mathcal{O}_{n_{\text{reg}}}^{\text{reg}}$ gives

$$\mathcal{O}^{\text{reg}} := \bigcup_{i=1}^{n_{\text{reg}}} \mathcal{O}_i^{\text{reg}}. \tag{4.23}$$

Dummy identity ontologies. Furthermore, we assume that ontologies we refer to as *dummy identity ontologies* be available. A dummy identity ontology specifies one or more dummy ontologies through their specification formulae Φ^{dy} comprising formulae $\{\gamma_i^{\text{dy}}\}$, where a formula γ_i^{dy} makes statements about the type and certifier of a dummy identity of the identity type issued by the certifier. The values of those attributes are the same for all identities of a given type from the same certifier. For the remaining attributes of a dummy identity, no predicates are specified through the specification formula for the dummy identity. A dummy identity ontology is required for policy matching in the general setting of data requests comprising disjunctions.

Through proper computation of the term that is used to refer to the identity in the formula of the ontology, a dummy identity is always referred to using the same term in the representation in ontologies and renamed in data statements to be released like other identities, which is exploited for interoperability between parties and the definition of the semantics. For each dummy identity being specified, specification formulae for a corresponding identifier object and identity of its certifier are comprised in the sets $\Phi^{\text{idf/ont}}$ and $\Phi^{\text{id/ont}}$, respectively, of the party specification ontology.

A dummy identity ontology can be created by an ontology provider by collecting public keys and certificate structures for identity types and certifiers of its interest, deriving the specification formulae, and formalizing multiple of them as a dummy identity ontology. For facilitating system openness, we do not rely on a situation that for all referred-to identity types in a formal deduction procedure it is mandated that an identity be specified in one of the party's identity relationships or be specified as a dummy identity in the ontology of the party. Rather, the party can obtain, for identity types referred to in ontologies being used and not being defined in the ontology it composes using dummy identity ontologies from third parties, the information required for constructing a dummy identity ontology on its own to accommodate reasoning over identities of these types. The latter is made explicit through the set $\widehat{\Phi}^{\text{dy}}$ in the discussions in Sec. 5.4 on the data release protocol based on credentials.³⁶ The formulae specifying dummy identities are required for finding fulfilling formulae for data requests while supporting disjunctions and the use of identity types of which the party does not have any identities in identity relationships of its portfolio.

For allowing for unconstrained reasoning over identities issued by the certifiers referred to in the party specification ontologies a party uses, specification formulae for dummy identities should be available that at least cover the union of all identities referred to in those ontologies. Due to their assessment of certifiers, providers of party assessment ontologies are likely to have the data required for defining those corresponding dummy identity ontologies. From a system perspective, having those parties issue both those kinds of ontologies is our approach of choice and is assumed to hold for practical systems. Taking this approach of mutually aligned party specification and dummy identity ontologies thus resolves the problem of a party being required to obtain corresponding dummy identities through other channels. Let $\mathcal{O}_1^{\text{dy}}, \dots, \mathcal{O}_{n_{\text{dy}}}^{\text{dy}}$ be the dummy identity ontologies a party uses. Then, its composed dummy identity ontology is defined through set union as

$$\mathcal{O}^{\text{dy}} := \bigcup_{i=1}^{n_{\text{dy}}} \mathcal{O}_i^{\text{dy}}.$$

4.10.2 Composition of Ontologies

In a practical open system, a party will be required to compose multiple ontologies to use for its reasoning in order to benefit from the advantages that the use of ontologies bring about in terms of expressivity and trust outsourcing. Composing $n_{\mathcal{O}}$ ontologies $\mathcal{O}_1, \dots, \mathcal{O}_{n_{\mathcal{O}}}$ of either kind as discussed earlier through set union results in a single ontology \mathcal{O} :

$$\mathcal{O} := \bigcup_{i=1}^{n_{\mathcal{O}}} \mathcal{O}_i.$$

³⁶From an implementation architecture perspective, $\widehat{\Phi}^{\text{dy}}$ can also be realized as a dummy identity ontology the party creates itself instead of obtaining it from an ontology provider as then this knowledge can be processed like third-party-obtained dummy identity ontologies.

The *compound ontology* \mathcal{O} comprises a key part of the theory \mathfrak{T} underlying the reasoning for deciding $\Gamma \vdash_{\mathcal{O}} \psi$, that is, deciding whether given premises Γ fulfill a formula ψ . This notation is equivalent to using the following sequent in the same derivative system: $\Gamma, \mathcal{O}_1, \dots, \mathcal{O}_{n_{\mathcal{O}}} \vdash \psi$. The formulae of the individual ontologies are assumed to hold as part of the premises for the deduction to be made.

As outlined already, composing ontologies may lead to unpredictable or undesired properties of the resulting compound ontology, particularly if ontologies being composed refer to or define the same concepts.

It is important to note that combining multiple ontologies where each of those may refer to the same concepts and make (slightly) different statements, may lead to an ontology that makes different statements about the same element, e.g., party. For example, the trust level of a certifier may be represented differently in two ontologies that are being combined and both assessments can be derived in the reasoning. This can make sense semantically, reflecting the different assessment procedures used.

The party using ontologies is responsible for a selection of proper ontologies and thus the theory resulting from those ontologies once composed. Clearly, it would counter the intention of using ontologies if parties using them need to scrutinize and fully understand them, which is particularly true for users and small service providers who do not have the required resources and knowledge. We next outline approaches for mitigating problems resulting from composing ontologies, without requiring that the party using the ontologies scrutinize or assess them.

Meta ontologies. The theory $\mathfrak{T}^{\mathcal{O}}$ resulting from using an ontology \mathcal{O} in a deduction, composed from multiple ontologies, must fulfill what the party intends to use as foundation for the theory for computing its derivations. As most parties will presumably lack the resources and technical capabilities of making such assessments, one solution thereto is to have independent organizations perform such assessments and publish machine-processable information on mutually compatible ontologies. The result is a *meta ontology* making statements on which ontologies may be used safely together for deductions. A meta ontology itself is, like other metadata, outside of the data representation formalism of our data model.

This approach permits multiple such meta ontologies by different organizations be made available. Each party in the system can select to trust one of those to rely on for the ontologies to use in its automated deductions. For this to work, it needs to be assumed that assessments of ontologies reflected in those meta ontologies be sufficiently close to each other such that parties relying on different meta ontologies obtain sufficiently similar and thus mutually compatible compound ontologies. Parties may also select to choose their own or no third-party ontologies, with the inherent loss of expressive power with respect to their policy specification, in case their application case permits so.

Market-driven selection. In an open system with multiple (competing) instances of ontologies being available for each kind of ontology, it can be expected that some of those ontologies will be adopted by a large fraction of the parties, based

on a *selection through market forces*. For example, quality of assessment, coverage, and reputation of the ontology providers can play a role in this. Assuming that ontologies in general model a well-delimited part of the system, where this part is delimited through conceptually meaningful parameters, different prominent ontologies of a certain type could be merged without the compositional problems discussed above arising.

4.10.3 Trust in Ontologies

Generalizing the approach of the assessment of parties taken for party specification ontologies leads to the trust assessment of ontology providers. As proposed in a report by Hogben and the author [HS06], we propose that related to the perceived value of the interaction being performed by a party, different ontologies, depending on the trust in their issuers, may be applied for reasoning related to this interaction. This facilitates security on the one hand and openness on the other hand, and allows for balancing those properties. This is an advanced topic and may allow for the use of low-trust ontologies for low-value transactions and high-trust ontologies for high-value transactions.

As hinted already earlier, when applying ontologie for making automated deductions in our logic, we may differentiate between trusted and untrusted ontologies. *Trusted ontologies* are obtained by a party from ontology providers it trusts for the purpose of providing ontologies of the type it obtains from it. All the kinds of ontologies we define, except for the identity binding ontologies, belong to this category. *Untrusted ontologies* are obtained from parties that are not trusted for the purpose of providing any kind of ontologies. This comprises certifiers—in an open system, every party can act as certifier—of identities in general, and thus makes the identity binding ontologies they issue untrusted.

Note, though, that trust is, in general, not a binary property and different degrees of trust apply to different providers. When only differentiating between trusted and untrusted ontologies instead of considering a spectrum of trust over ontologies [HS06], trusted ontologies can be composed with the ontology forming the theory for reasoning, while untrusted ontologies need to be verified or derived from a representation that ensures it not influencing the reasoning in an adversarial manner, before composing it with the trusted ontologies.

4.10.4 Interoperability Challenges

An open ontology approach for an identity management system poses multiple interoperability challenges. Assuming an open system, it is likely that two interacting parties will use different sets of ontologies to compose their respective ontology to make derivations under, that is, different theories. For concept specification or party assessment ontologies this may mean that the parties use similar, though not necessarily the same, *vocabulary*, as well as similar, though not necessarily identical, *assessments* of parties.

Regarding the vocabulary, it is crucial that the concepts both parties need to refer to, e.g., in a data request and a corresponding response data statement, are the same, that is, that the ontologies have an intersection with respect to this. Though, the assessment of parties by different assessors may deviate to some degree, e.g., through different assessment procedures used by the ontology providers or judgment employed in some steps of the assessment procedure. Expressed differently, the composed ontologies used by the interacting parties need to be sufficiently “close” to each other.

The parties in an interaction using different, but close, ontologies can result in the same properties or (slightly) different properties derived through automated reasoning in an instance of deduction. Both parties obtaining sufficiently close results is a prerequisite for the interaction to work, e.g., a user being able to properly fulfill a data request issued by a service provider. Particularly, policies should be flexible enough such that they can still be fulfilled by other parties using slightly different ontologies than the policy authoring party as this is key for interoperability of an open system based on our ontology framework.

Example 4.28 (Closeness of ontologies) *Take the following as an example for the closeness of the ontologies of interaction partners and flexible specification of the policy. Ontology provider P_1 assesses certifier C and assigns a trust level of 3 to it, while provider P_2 assesses the trust level of C to be 4. Data recipient (relying party) R requires a trust level of at least 3 for access to one of its services and states this in an access control policy. A user U can use an identity issued by C to authorize at R when using P_1 ’s ontology, while R checks fulfillment of the policy using the ontology of P_2 .*

As assessment procedures are not perfect, this approach may sometimes also lead to a failure of interactions due to ontologies diverging too much or policies being specified in a “too tight” manner with respect to the party authoring the policy, that is, too close to the assessment specified in the ontology. Different ontologies represent different views on the system and its parties of different trust providers and thus having different ontologies reflects the real-world situation of assessment differences, and particularly also judgment and opinions may play a role in the assessment procedures. For example, a service provider’s policy reflects this party’s world or system view and a user’s identity statement constructed in response to such policy reflects the user’s world or system view, both based on the ontologies those parties use in their processing. In the case when the ontologies of the parties in an authentication interaction are not sufficiently close, this may result in denial of access, even though in the view of the authenticating party the authentication should have succeeded. That is, it can lead to an excessive release of data without the expected gain, but it cannot lead to obtaining access the authenticating party is not eligible for because the server always uses its trusted ontologies, that is, its world view, for the access decision. The imperfections outlined above are, in our opinion, the natural consequence of an open ontology-based identity system without central authorities.

Using appropriate namespaces for defining the vocabulary referred to in an ontology is crucial for avoiding unintended interrelations between two ontologies. That is, when a provider defines a new vocabulary, it must use terms from a namespace that ensures collision freeness with terms chosen by other providers. The URI scheme [BLFM05] by W3C is an appropriate scheme for this, as discussed for a related interoperability problem in the scope of the overall data model in Sec. 4.5.2.

4.10.5 Distribution of Ontologies

One main requirement for identity management of the kind we propose is that an architecture and infrastructure for the handling of ontologies, including their secure, that is, integrity protecting, distribution, be in place. This architecture is the same for the different kinds of ontologies we use. Protocols for obtaining ontologies need to ensure the *integrity* of the ontology. This can be technically achieved through applying standard (cryptographic) mechanisms, such as standard PKI-based signatures by the ontology provider over the canonicalized representation of the ontology, analogous to certifying public keys through public key certificates.

4.10.6 Discussion

The use of ontologies is *optional* to a certain extent in the sense that a basic deployment of our system can operate without ontologies provided by third parties, of course implying a substantial loss in the expressive power and openness of our identity system. For bootstrapping a system, a first deployment phase can be done with fewer or no third-party ontologies being available while still providing the basic identity services for transferring authentications between the system participants. At a later stage, ontology-providing parties can emerge and offer additional value to the system participants through the ontologies they issue. For example, such ontology providers can assess third parties such as certifiers or trustees. Providers that issue subtype hierarchies for identity types could be related to the issuers, e.g., being an organization involved in defining a type hierarchy over European electronic identity credentials.

One of the more substantial non-technical issues related to third-party-issued ontologies is *liability*, e.g., liability for the assessments made by an ontology provider, or the rules provided by it. The liability issues get considerably more elaborate when using a composed ontology for reasoning, particularly for the case of unintended consequences introduced through the composition. A discussion of those aspects is left to future work, closely related to the technical aspects of ontology composition. Those discussions need to be generalized in the case of using meta ontologies.

The use of ontologies in an identity management system increases the overall system complexity, in both technical and non-technical dimensions, substantially, though is required for obtaining an open system. Multiple aspects related to ontologies have been touched only in our discussions, while a more detailed elaboration is left to future work. Our main contributions with respect to ontologies are the tight integration of ontologies into our system-wide data model, thus also integration with

the deduction system, policy matching process, and thus all processes for realizing the release of certified data, and the discussion of various challenges and proposal of solutions.

4.11 Semantics

A formal language on its own does not have meaning because of only being expressed at the syntactic level, that is, formulae expressed in it merely are compositions of strings. The meaning of formulae is given by the semantics and the deductive system of the logic. The model-theoretic, also known as Tarski-style, or denotational, semantics of a formal language as considered in this thesis is concerned with the denotations, **true** or **false**, of the sentences over the language [Gir90]. This is achieved through interpretations, that is, assigning objects from a universe of discourse to the terms of the language, which gives meaning to the formulae. A discussion on the deduction system is given in Sec. 4.9.2 on the logic.

Only statement-type formulae, that is, such without free variables, can be interpreted as they do not comprise free variables and thus get a denotation assigned through the semantics. The different dialects of the data statement language for different technologies as well as object specification formulae are statement-type formulae.

An *interpretation* \mathcal{I} is a mapping from constant terms to elements of the domain of discourse and from function and relation symbols to concrete functions and relations.

The *domain of discourse* is the set that the symbols of the language are mapped to and means, intuitively speaking, what the language being interpreted is about, that is, the set of elements that the terms of the language refer to. The domain of discourse is also referred to as domain, or universe of discourse, or universe. Because we operate on a sorted logic, the domain of discourse comprises a set of elements for each of the sorts defined in the language.

Recall that the type definitions for the data types can be seen as part of an *ontology* or type system \mathcal{T} which defines those types with the operators and relations specified on them and the interpretations thereof. That is, those definitions are fixed for all interpretations and need not be redefined in each interpretation. Notation-wise, though, we handle all terms equally in the following discussions, regardless of \mathcal{T} . Handling the above aspects in the type system may be relevant for the logic retaining its first-order properties.

The semantics defines, for a complete system, the meaning of the statement-type formulae expressed in this system in a specific state \mathcal{Q}^δ , where a state is defined through the protocols of Chapter 5 executed so far. Thus, it provides the formal logic foundation for the cryptographic protocols. Also, it allows a party who has observed or participated in a subset of the protocols up to this state to verify the denotation of data statements it has received being equal to **true** through verification of the protocols.

The matching of a party's portfolio against a formula specifying a data request

as explained in Sec. 5.4.6 is done solely on the syntactic level, without account to an interpretation. The utility of the semantics in this context is to provide the underlying interpretation reflecting the current system, thereby giving meaning to the operation of matching.

4.11.1 Types and Interpretation Domains

For the definition of an interpretation \mathcal{I} for our language, we build upon the concept of a *system state* or *state*. A state is defined through the sequence of all instances of cryptographic protocols that have been executed. Thus, it determines the existing identities, conditionally released identities, opaque identities, and identifier objects. Particularly, it also determines the terms used to refer to those, also considering the use of different terms to refer to a single such object.

We assume a discrete state transition system with states being in a total order. Furthermore, we assume that each transition corresponds to a discrete time value. We denote the system state at a given time δ as \mathcal{G}^δ . Executing a cryptographic protocol induces a transition of the system to a new system state $\mathcal{G}^{\delta+1}$.

We make a distinction between the syntactically valid terms for a type $\tilde{\tau}$ as specified through its grammar and its state-dependent specification $\tilde{\tau}^{\mathcal{G}^\delta}$ in order to have the basic language specification through the grammar independent of the system state, while ensuring that only terms for which a mapping is defined in the interpretation function are considered for a given interpretation. Thus, such type is given by the system state \mathcal{G}^δ based on which the interpretation is to be done. Formulae that comprise terms not defined in a given system state do not have an interpretation, although if they are well-formed according to the syntax.

Types $\tilde{\tau} \in \{\text{idfo}, \text{id}, \text{crid}, \text{oid}, \text{idf}, \text{pty}\}$ have a state-dependent type extension $\tilde{\tau}^{\mathcal{G}^\delta}$ and thus depend on the state \mathcal{G}^δ the interpretation is to be done in. Their state-dependent specifications are referred to as $\tilde{\tau}^{\mathcal{G}^\delta}$. For simplifying the notation, we may omit the system state from the type when it is clear which state we mean.

For our typed logic, an interpretation requires that each type $\tilde{\tau}$ be mapped to a domain $\mathfrak{D}_{\tilde{\tau}^{\mathcal{G}^\delta}}$, the *interpretation domain* of the type. We may again simplify notation with respect to state-dependent type specifications and use $\mathfrak{D}_{\tilde{\tau}}$ when it is clear what is referred to. Any set can have its corresponding interpretation domain in a given state of the system, expressed using similar notation.

The interpretation domain for a type can be independent of \mathcal{G}^δ and identical to the type's extension with each element being mapped to itself. For the latter we say that the terms of the type are *self referential* in an interpretation. This is the case for the data types in \mathcal{T}^{dat} , that is, $\{\text{int}, \text{str}, \text{bool}, \text{date}\}$ and $\{\text{date}^\xi : \xi \in \{\text{Y}, \text{M}, \text{D}, \text{h}, \text{m}, \text{s}\}\}$ defined for our system and the type att for attribute names.

For the types idfo , id , crid , oid , and pty , the respective interpretation domain is different to the set induced by the type. The first four of the mentioned types have functions as elements of their interpretation domains as induced through the executed protocols modeling the creation of the identifier object, identity, conditionally released identity, and opaque identity, respectively. The type idf maps to the interpretation domain of identifiers existing for established identifier objects. The

type **pty** maps to the interpretation domain comprising all parties in the system. The types **id** and **pty** have in common that multiple elements of the type can map to a single element of the interpretation domain through an interpretation function.

Due to the property of our logic of not being based on the unique names assumption, the following holds for those types and interpretation domains for $\tilde{\tau} \in \{\mathbf{id}, \mathbf{pty}\}$:

$$|\tilde{\tau}^{\mathcal{G}^\delta}| \geq |\mathfrak{D}_{\tilde{\tau}^{\mathcal{G}^\delta}}|.$$

4.11.2 Interpretation

We model an identity $\mathbf{c} :: \mathbf{id}$ of identity type τ in the system as a function which maps an attribute name for an attribute of type β to the corresponding attribute value of this type for all attributes defined by the identity type τ , where $\mathcal{A}_\tau = \{a_1, \dots, a_{k_{\beta_\tau}}\}$ with $a_i \in \llbracket \mathbf{att} \rrbracket$ is the set of attributes of the identity type and $\mathcal{T}_\tau^{\mathbf{id}}$ the set of data types corresponding to the attributes. Such function is represented through a function symbol \mathbf{c} defined as

$$\mathbf{c} : \mathcal{A} \rightarrow \bigcup_{\beta_i \in \mathcal{T}_\tau^{\mathbf{id}}} \llbracket \beta_i \rrbracket,$$

with the property that $\mathbf{c}(a_i) :: \beta_i$.

Note that identities and other objects, such as identifier objects, opaque identities, and conditionally released identities, are technically records, similar to records in programming languages. However, we treat identities and such other objects as functions for the convenience of the resulting notation and terminology. Those functions have finite domains and thus can be modeled as elements of the universe of discourse. It is easy to see that the typing of such records can be done axiomatically and that it can be expressed easily in classical first-order logic. Henceforth, we will use the function notation and terminology to refer to identities and other objects.

Using the symbol \mathbf{c} , the prominently used notation $\mathbf{c}.a$ for referring to attribute a of identity \mathbf{c} is thus only an alternative notation in our language for the function notation $\mathbf{c}(a)$. We use this alternative notation for it being, thanks to its record syntax, more intuitive, though, we need to revert to the formal definition in the current discussion of the semantics and for being aligned with first-order logic concepts. Analogous to identities as explained, all conditionally released identities, opaque identities, and identifier objects correspond to functions.

$\mathfrak{D}_{\mathbf{id}}$ comprises the functions representing identities existing in the system at state \mathcal{G}^δ . The function $\mathcal{X}_\mathbf{c}^{\mathbf{id}}$ is such function for identity \mathbf{c} of type τ , where $\mathcal{A} = \{a_1, \dots, a_{k_{\beta_\tau}}\}$ is the set comprising attribute names, k_{β_τ} the number of attributes of τ , $\mathcal{T}^{\mathbf{id}}$ the corresponding set of data types, and β_i one of the data types:

$$\mathcal{X}_\mathbf{c}^{\mathbf{id}} : \mathfrak{D}_\mathcal{A} \rightarrow \bigcup_{\beta \in \mathcal{T}_\tau^{\mathbf{id}}} \mathfrak{D}_{\beta_i}.$$

As the type **att** is self-referential, $\mathfrak{D}_\mathcal{A}$ is equal to $\mathcal{A} \subseteq \llbracket \mathbf{att} \rrbracket$. The interpretation domains for the types **idfo**, **crid**, and **oid** for identifier objects, conditionally released

identities, and opaque identities, respectively, comprise the respective functions resembling the objects of the respective object types. The function corresponding to an identifier object p is $\mathcal{X}_p^{\text{idf}}$, that of a conditionally released identity e is $\mathcal{X}_e^{\text{crid}}$, and that of an opaque identity q is $\mathcal{X}_q^{\text{oid}}$.

For a constant or basic variable t of type β , that is, $t :: \beta$, its interpretation, denoted as $t^{\mathcal{I}}$, is an element of \mathfrak{D}_β , that is, $t^{\mathcal{I}} \in \mathfrak{D}_\beta$. For an identity $c \in \llbracket \text{id} \rrbracket$, we have $c^{\mathcal{I}}$ being a concrete function $\mathcal{X}_c^{\text{idv}} \in \mathfrak{D}_{\text{id}}$. For an identity c of a type τ , that is, $c \in \llbracket \tau \rrbracket$, $c^{\mathcal{I}}$ is a function $\mathcal{X}_c^{\text{idv}} \in \mathfrak{D}_\tau \subseteq \mathfrak{D}_{\text{id}}$. The interpretation of an attribute reference $c.a$ of an identity is defined as

$$(c.a)^{\mathcal{I}} = (c^{\mathcal{I}})(a^{\mathcal{I}}),$$

which is formally equivalent to the function notation $c^{\mathcal{I}}(a^{\mathcal{I}})$. The interpretations for other objects and their attributes are analogous to the interpretation for identities.

The interpretation of an attribute reference can be extended recursively to expressions built using operators (functions) \circ of *DefOps*, which defines the operators over the basic data types, as defined through \mathcal{T} on arguments of type $\beta = \text{int}$:

$$(s :: \beta \circ t :: \beta)^{\mathcal{I}} = s^{\mathcal{I}} \circ_\beta t^{\mathcal{I}}.$$

The type β is the type of the subexpressions s and t , where the type of both subexpressions must be the same and the operator \circ_β must be defined according to our type system \mathcal{T} .

For date types, the situation is slightly different, as for operations on dates one argument is a date while the other one is a date duration. No recursive definition is required due to the simplified syntax of date expressions. Let $s :: \beta, t :: \text{int}$, where $\beta \in \{\text{date}, \text{date}^Y, \dots, \text{date}^S\}$, and let date expressions be defined as follows:

$$(s :: \beta \circ t :: \text{int})^{\mathcal{I}} = s^{\mathcal{I}} \circ_\beta t^{\mathcal{I}}.$$

Other types than int and the date types do not have operators defined.

A predicate symbol R with arguments t_1, \dots, t_k is interpreted as follows through interpretation of its arguments and the predicate symbol with a relation:

$$R(t_1, \dots, t_k)^{\mathcal{I}} = R^{\mathcal{I}}(t_1^{\mathcal{I}}, \dots, t_k^{\mathcal{I}}).$$

A function symbol f with arguments t_1, \dots, t_k is interpreted as follows, again by interpreting its arguments and the function symbol with a function:³⁷

$$f(t_1, \dots, t_k)^{\mathcal{I}} = f^{\mathcal{I}}(t_1^{\mathcal{I}}, \dots, t_k^{\mathcal{I}}).$$

Free variables, that is, such not captured in the scope of quantifiers, are handled by requiring that they be mapped to constant terms. For this, we use an *environment* \mathcal{E} , also called *variable assignment* or *assignment*, which is a function from the free variables of a formula to terms of the language. A formula comprising free variables can only be interpreted with respect to an environment \mathcal{E} that maps all free variables to terms which we can interpret.

The object identity $t_1 = t_2$ is built into the logic and has the semantics that both terms connected with the $=$ -symbol refer to the same object in any interpretation.

³⁷Note again our use of the function terminology in this work.

Example: Expressions. The interpretation of an expression exp is defined recursively via the interpretations of the components it comprises as discussed above. Both the type and the value of an expression are defined through its components. Next, we present the interpretations for expressions of types **int** and **date** as examples.

Expression terms of type **int** are defined as a sum of l products of constant factors u_{j_i} and attribute references $o_{j_i}.a_{k_i}$ as well as a constant value:

$$exp = u_{j_1} \cdot o_{j_1}.a_{k_1} + \dots + u_{j_i} \cdot o_{j_i}.a_{k_i} + \dots + u_{j_l} \cdot o_{j_l}.a_{k_l} + u_{j_{l+1}} \cdot$$

For the interpretation $exp^{\mathcal{I}}$ of exp , an integer constant u_{j_i} is interpreted with itself and the attribute reference is interpreted in the standard way through a function. The operator symbols $+$ and \cdot which are function symbols in the language are replaced with the corresponding group operations $+_{\text{int}}$ and \cdot_{int} over the integers. We may omit the $_{\text{int}}$ -decoration for those operators for notational simplicity.

$$exp^{\mathcal{I}} = u_{j_1}^{\mathcal{I}} \cdot (o_{j_1}.a_{k_1})^{\mathcal{I}} + \dots + u_{j_i}^{\mathcal{I}} \cdot (o_{j_i}.a_{k_i})^{\mathcal{I}} + \dots + u_{j_l}^{\mathcal{I}} \cdot (o_{j_l}.a_{k_l})^{\mathcal{I}} + (u_{j_{l+1}})^{\mathcal{I}}$$

A date expression is much more constrained than an integer expression and its interpretation is defined analogously. Next, we present the kind of date expressions with the first argument being an attribute reference $o.a :: \text{date}^{\mathcal{E}}$ and the second argument an integer $u :: \text{int}$, while other forms of such expressions exist, e.g., one that has a date constant as its first argument, with accordingly simpler interpretation:

$$exp = o.a + u$$

Such an expression is interpreted analogous to an integer expression by interpreting both arguments and the operator symbol following the rules established further above.

$$exp^{\mathcal{I}} = (o.a)^{\mathcal{I}} + u^{\mathcal{I}}$$

We define the semantics of a statement ϕ using a standard inductive definition of a models-relation:

$$\begin{aligned} \mathcal{I} &\models \phi_1 \wedge \phi_2 \text{ iff } \mathcal{I} \models \phi_1 \text{ and } \mathcal{I} \models \phi_2 \\ \mathcal{I} &\models \phi_1 \vee \phi_2 \text{ iff } \mathcal{I} \models \phi_1 \text{ or } \mathcal{I} \models \phi_2 \\ \mathcal{I} &\models \neg\phi \text{ iff } \mathcal{I} \not\models \phi \\ \mathcal{I} &\models \exists x : \phi \text{ iff } x :: \beta \text{ and there is } c \in \llbracket \beta \rrbracket \text{ such that } \mathcal{I}[x \mapsto c] \models \phi \\ \mathcal{I} &\models t = s \text{ iff } t^{\mathcal{I}} = s^{\mathcal{I}} \\ \mathcal{I} &\models \forall\phi \text{ iff } \mathcal{I} \models \phi \text{ for all } \phi \\ \mathcal{I} &\models P(t_1, \dots, t_k) \text{ iff } P^{\mathcal{I}}(t_1^{\mathcal{I}}, \dots, t_k^{\mathcal{I}}) \end{aligned}$$

Interpretability of formulae comprising disjunctions. Formulae with disjunctions deserve specific mention as for their interpretation we need to ensure that a function $\mathcal{X}_c^{\text{id}}$ exist for every identity function symbol c referred to therein. For other objects, a function exists by definition of the protocols, even though it may be created only in the process of proving the formula. The following two cases at least require consideration of this: (1) A party who intends to reveal and prove correctness of such formula comprising a disjunction to another party, and (2) a party who establishes an identity relationship which comprises a disjunction.

In either of the above cases, we cannot yet ensure, for a referred-to identity $c :: \tau$ for which the predicates stated in the formula do not necessarily hold, that a function $\mathcal{X}_c^{\text{id}} \in \mathfrak{D}_\tau \subseteq \mathfrak{D}_{\text{id}}$ exists. This is due to the use of a disjunction in the formula, that is, only one of the branches of the disjunction needs to be fulfilled by corresponding identities of the party. From this it follows that it is not ensured that an interpretation of such formula exists, although they are syntactically perfectly valid. The problem with the interpretation can be solved by introducing the concept of *dummy identities*.

Definition 4.3 (Dummy identity) *A dummy identity c for an identity type τ comprises the type and certifier attribute reflecting the type τ and a certifier for this type, formalized through the corresponding function $\mathcal{X}_c^{\text{id}}$. For all further attributes of τ , the dummy identity comprises a special, reserved, value.³⁸*

A dummy identity $c_\tau :: \tau$ is defined for each pair of static identity type τ and certifier and can be referred to in formulae like any other identity. Note that $\mathcal{X}_c^{\text{id}} \in \mathfrak{D}_\tau$ and availability of such function for each dummy identity ensures that formulae with disjunctions can be interpreted in general. Also note that the special values for other attributes than the type and certifier are required for type correctness of formulae. Further details on dummy identities in a system are given in Sec. 5.1.2.

Note that Case (2) cannot be realized with our credential protocols, though, can be realized with protocols based on online certifiers who vouch for the statement in an identity relationship.

Well-typed formulae. The semantics of well-typed statement-type formulae has been defined above. The semantics of non-well-typed, or ill-typed, formulae is undefined as it is unclear what the meaning of such statements is in reality in this case. One example for an ill-typed statement is a formula comprising a predicate with arguments of types not compliant with the predicate signature, e.g., the predicate $\text{Eq}_{\text{str} \times \text{str}}(c.\text{lastName}, t)$ with $\text{lastName} :: \text{str}, t :: \text{int}$.

Recall that, due to our approach of handling the typing of identities as presented in Sec. 4.4.11.2, all identities are of the same type id without subtyping through the built-in typing of the logic. In contrast to related work [CMN⁺10], we can therefore not ensure only through considering the built-in types of terms for the typing of identities, conditionally released identities, opaque identities, and identifier objects

³⁸Note that we do not further define the treatment of this special value per data type in the respective sections on formal languages and semantics.

that the interpretation of attribute references of such objects is defined in case a formula violates our static typing scheme for objects. In such type violation case, the function image corresponding to the identity for a referenced attribute is undefined. In the case that a function cannot be evaluated, there exists no interpretation for a given formula and the formula is considered to not be a well-formed formula of our logic.

The reason for this less stringent typing system of the logic itself is that we do not define identity types in the standard type system of the logic, but on top of it. This holds analogously for opaque and conditionally released identities. This is a minor drawback of our modeling approach for the typing of those objects.

4.11.3 Predicates

We next define the semantics of the predicate symbols of the language through the interpretation of their arguments and concrete predicates.

Note that the technical implementation of the predicates, e.g., encoding of date values as integers, does not need to be considered in the semantics of the language and needs to be handled in the implementation based on private certificate protocols.

Predicates on integers. Let the predicate symbol $P_{\text{int} \times \text{int}}$ be any of the symbols referring to relation predicates on arguments of type int :

$$P_{\text{int} \times \text{int}} \in \{\text{Eq}_{\text{int} \times \text{int}}, \text{Neq}_{\text{int} \times \text{int}}, \text{Lt}_{\text{int} \times \text{int}}, \text{Leq}_{\text{int} \times \text{int}}, \text{Geq}_{\text{int} \times \text{int}}, \text{Gt}_{\text{int} \times \text{int}}\}.$$

The semantics of a predicate $P_{\text{int} \times \text{int}}$ over the integers can be defined through interpreting its arguments and interpreting the predicate symbol with the corresponding relation over the integers:

$$\mathcal{I} \models P_{\text{int} \times \text{int}}(\text{exp}_1 :: \text{int}, \text{exp}_2 :: \text{int}) \text{ iff } P_{\text{int} \times \text{int}}^{\mathcal{I}}(\text{exp}_1^{\mathcal{I}}, \text{exp}_2^{\mathcal{I}})$$

$P_{\text{int} \times \text{int}}^{\mathcal{I}}$, the interpretation of $P_{\text{int} \times \text{int}}$, is defined as the corresponding relation over the integers as

$$P_{\text{int} \times \text{int}}^{\mathcal{I}} = \begin{cases} =_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Eq}_{\text{int} \times \text{int}} \\ \neq_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Neq}_{\text{int} \times \text{int}} \\ <_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Lt}_{\text{int} \times \text{int}} \\ \leq_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Leq}_{\text{int} \times \text{int}} \\ \geq_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Geq}_{\text{int} \times \text{int}} \\ >_{\text{int}} & \text{if } P_{\text{int} \times \text{int}} = \text{Gt}_{\text{int} \times \text{int}} \end{cases},$$

where the relations $=_{\text{int}}$, \neq_{int} , $<_{\text{int}}$, \leq_{int} , \geq_{int} , and $>_{\text{int}}$ are the standard equality, non-equality, and inequality relations over the integers.³⁹

³⁹For notational convenience we abbreviate the typing decoration $\beta \times \beta$ of the binary relations with a single β .

Note that the infix notation $t_1^{\mathcal{I}} P_{\text{int} \times \text{int}}^{\mathcal{I}} t_2^{\mathcal{I}}$ for a binary relation on terms $t_1^{\mathcal{I}}, t_2^{\mathcal{I}}$ for its interpretation is equivalent to its relation notation $P_{\text{int} \times \text{int}}^{\mathcal{I}}(t_1^{\mathcal{I}}, t_2^{\mathcal{I}})$ and we may use those notations interchangeably. Note also that the arguments of the predicate $P_{\text{int} \times \text{int}}$ are expressions over the integers and thus integers in the trivial case.

Predicates on dates. The various date types are particularly interesting because of the generic type **date** and the different granularized types date^ξ that may be arguments of predicates.

As a prerequisite, we define a function $\text{Expand}(d, \xi_2)$ to increase the granularity of a given date constant $d :: \xi_1$ to granularity ξ_2 by concatenating delimiting tokens and zeroes to the right side of the representation of the constant d until it is expressed with granularity ξ_2 . Let $\widehat{\mathcal{G}}^{\text{date}} := \{(Y, M), (M, D), (D, h), (h, m), (m, s)\} \subset \mathcal{G}^{\text{date}}$ be the ordering expressed over all “adjacent” symbols indicating date granularities. Let $\mathcal{G}_{\text{TO}}^{\text{date}} := \{(1, Y), (2, M), (3, D), (4, h), (5, m), (6, s)\}$ define a total order over the date granularities, ranging from coarse to fine granular. In 4.24, we define the Expand function through a recursive definition based on $\widehat{\mathcal{G}}^{\text{date}}$ and $\mathcal{G}_{\text{TO}}^{\text{date}}$.

$$\text{Expand}(d, \xi_2) := \begin{cases} d \parallel \text{"-00"} & \text{if } d :: \text{date}^Y, \xi_2 = M \\ d \parallel \text{"-00"} & \text{if } d :: \text{date}^M, \xi_2 = D \\ \dots & \\ d \parallel \text{"-00"} & \text{if } d :: \text{date}^m, \xi_2 = s \\ \text{Expand}(\text{Expand}(d, \xi'), \xi_2) & \text{if } d :: \text{date}^{\xi_1}, (i, \xi_1) \in \mathcal{G}_{\text{TO}}^{\text{date}}, \\ & (j, \xi') \in \mathcal{G}_{\text{TO}}^{\text{date}}, i < j, \\ & (\xi', \xi_2) \in \widehat{\mathcal{G}}^{\text{date}} \end{cases} \quad (4.24)$$

As an example, the function with arguments $\text{Expand}(2011-07-01, s)$ evaluates to 2011-07-01-00-00-00 in granularity s indicating seconds.

Next, we present the predicates on date arguments of the same granularized date type date^ξ with $\xi \in \{Y, M, D, h, m, s\}$ as

$$\begin{aligned} \text{exp}_1, \text{exp}_2 &:: \text{date}^\xi \\ \mathcal{I} \models P_{\text{date}^\xi \times \text{date}^\xi}(\text{exp}_1, \text{exp}_2) &\text{ iff } P_{\text{date}^\xi \times \text{date}^\xi}^{\mathcal{I}}(\text{exp}_1^{\mathcal{I}}, \text{exp}_2^{\mathcal{I}}) \end{aligned}$$

with

$$P^{\mathcal{I}} = \begin{cases} =_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Eq}_{\text{date}^\xi \times \text{date}^\xi} \\ \neq_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Neq}_{\text{date}^\xi \times \text{date}^\xi} \\ <_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Lt}_{\text{date}^\xi \times \text{date}^\xi} \\ \leq_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Leq}_{\text{date}^\xi \times \text{date}^\xi} \\ \geq_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Geq}_{\text{date}^\xi \times \text{date}^\xi} \\ >_{\text{date}^\xi} & \text{if } P_{\text{date}^\xi \times \text{date}^\xi} = \text{Gt}_{\text{date}^\xi \times \text{date}^\xi} \end{cases},$$

where the relations $=_{\text{date}^\xi}, \neq_{\text{date}^\xi}, <_{\text{date}^\xi}, \leq_{\text{date}^\xi}, \geq_{\text{date}^\xi}$, and $>_{\text{date}^\xi}$ are the standard relations over the given date type.

$\text{Eq}_{\text{date} \times \text{date}}$ is the semantically most relaxed equality predicate with both arguments being typed with the generic date type **date** and therefore comprising a value of any of the granularized date types. It interprets the less-granular argument as a range and evaluates the range boundaries with respect to the other argument using predicates for arguments of the same granularity. Equality holds if the more fine-granular argument is within the range that the less-granular one is interpreted as. There are two cases to consider, namely the first or the second argument of the predicate being the more fine granular argument.

$$\mathcal{I} \models \text{Eq}_{\text{date} \times \text{date}}(d_1, d_2) \text{ iff } \left\{ \begin{array}{l} \mathcal{I} \models \text{Geq}_{\text{date}^{\xi_2} \times \text{date}^{\xi_2}}(d_2, \text{Expand}(d_1, \xi_2)) \text{ and} \\ \mathcal{I} \models \text{Lt}_{\text{date}^{\xi_2} \times \text{date}^{\xi_2}}(d_2, \text{Expand}(d_1 + 1, \xi_2)) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_1, \xi_2), \xi_1 \neq \xi_2 \\ \mathcal{I} \models \text{Geq}_{\text{date}^{\xi_2} \times \text{date}^{\xi_2}}(d_1, \text{Expand}(d_2, \xi_1)) \text{ and} \\ \mathcal{I} \models \text{Lt}_{\text{date}^{\xi_2} \times \text{date}^{\xi_2}}(d_1, \text{Expand}(d_2 + 1, \xi_1)) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_2, \xi_1), \xi_1 \neq \xi_2 \end{array} \right.$$

The above specifies the semantics of the predicate symbol $\text{Eq}_{\text{date} \times \text{date}}$ over the generic date type completely through using predicate symbols on the granularized types the semantics of which is defined as explained before and expressing the models-relation on those.

Non-equality $\text{Neq}_{\text{date}, \text{date}}^{\mathcal{I}}$ is the negation of the above and realized analogously.

Example 4.29 (Equality of arguments of different types of date) *A concrete example of arguments being of day and hour granularity and the definition of the predicate for such arguments follow:*

$$d_1 :: \text{date}^{\text{D}}, d_2 :: \text{date}^{\text{h}}$$

$$\mathcal{I} \models \text{Eq}(d_1, d_2) \text{ iff } \mathcal{I} \models \text{Geq}(d_2, \text{Expand}(d_1, \text{h})) \text{ and } \mathcal{I} \models \text{Lt}(d_2, \text{Expand}(d_1 + 1, \text{h})) .$$

Inequalities with arguments d_1, d_2 of type **date** with different granularities are defined through inequality predicates on the same granularized types. First, we present the definition of the predicate $\text{Leq}_{\text{date} \times \text{date}}$:

$$\mathcal{I} \models \text{Leq}_{\text{date} \times \text{date}}(d_1, d_2) \text{ iff } \left\{ \begin{array}{l} \mathcal{I} \models \text{Leq}_{\xi_2 \times \xi_2}(\text{Expand}(d_1, \xi_2), d_2) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_1, \xi_2), \xi_1 \neq \xi_2 \\ \mathcal{I} \models \text{Lt}_{\xi_1 \times \xi_1}(d_1, \text{Expand}(d_2, \xi_1)) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_2, \xi_1), \xi_1 \neq \xi_2 . \end{array} \right.$$

The inequality $\text{Lt}_{\text{date} \times \text{date}}$ is realized similar to the above:

$$\mathcal{I} \models \text{Lt}_{\text{date} \times \text{date}}(d_1, d_2) \text{ iff } \begin{cases} \mathcal{I} \models \text{Leq}_{\xi_2 \times \xi_2}(\text{Expand}(d_1 + 1, \xi_2), d_2) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_1, \xi_2), \xi_1 \neq \xi_2 \\ \mathcal{I} \models \text{Lt}_{\xi_1 \times \xi_1}(d_1, \text{Expand}(d_2, \xi_1)) \\ \quad \text{if } d_1 :: \text{date}^{\xi_1}, d_2 :: \text{date}^{\xi_2}, \\ \quad \mathcal{G}^{\text{date}}(\xi_2, \xi_1), \xi_1 \neq \xi_2 . \end{cases}$$

The further inequality relations $\text{Geq}_{\text{date} \times \text{date}}$ and $\text{Gt}_{\text{date} \times \text{date}}$ between arguments of different granularized types are handled analogously considering the range semantics of the less-granular argument.

Note that using our cryptographic credential protocols, only a subset of the defined predicates can be proven in zero knowledge depending on which of the arguments are constants or expressions.

Considering the above discussions, the support of different granularized types in the relation predicates substantially increases complexity of the semantics and implementation, and it particularly can lead to increased runtime for equality relations.

Predicates on strings. The predicates our language supports for strings are equality $\text{Eq}_{\text{str} \times \text{str}}$ and the negation $\text{Neq}_{\text{str} \times \text{str}}$ thereof, while predicates for expressing lexicographic ordering over strings are not supported due to the performance issues of the related cryptographic protocols. The semantics for the predicate $\text{Eq}_{\text{str} \times \text{str}}$ on arguments of type **str** is defined as follows, where $t_1, t_2 :: \text{str}$ and $=_{\text{str}}$ is the relation for string equality for elements of the interpretation domain $\mathfrak{D}_{\text{str}}$:

$$\mathcal{I} \models \text{Eq}_{\text{str} \times \text{str}}(t_1, t_2) \text{ iff } t_1^{\mathcal{I}} =_{\text{str}} t_2^{\mathcal{I}} .$$

The semantics of negation $\text{Neq}_{\text{str} \times \text{str}}$ of the $\text{Eq}_{\text{str} \times \text{str}}$ predicate can be defined directly through the non-equality \neq_{str} of the interpretations of the string arguments through the \neq_{str} -operator for arguments of the interpretation domain:

$$\mathcal{I} \models \text{Neq}_{\text{str} \times \text{str}}(t_1, t_2) \text{ iff } t_1^{\mathcal{I}} \neq_{\text{str}} t_2^{\mathcal{I}} .$$

Predicates on booleans. The semantics for the type **bool** is easy to define following the ideas given already above for **str**-typed arguments:

$$\begin{aligned} \mathcal{I} \models \text{Eq}_{\text{bool} \times \text{bool}}(t_1, t_2) &\text{ iff } t_1^{\mathcal{I}} =_{\text{bool}} t_2^{\mathcal{I}} , \\ \mathcal{I} \models \text{Neq}_{\text{bool} \times \text{bool}}(t_1, t_2) &\text{ iff } t_1^{\mathcal{I}} \neq_{\text{bool}} t_2^{\mathcal{I}} . \end{aligned}$$

Equality on party identifiers. A data type deserving specific mention is the type **pty** as different terms can be interpreted with the same identifier for a party. This is a key property of our approach to data representation and impacts the semantics: Only equality of parties can be expressed in our logic using object equality

(=). The predicate is true if both its arguments are interpreted with the same party in an interpretation \mathcal{I} .

Equality over parties is defined through equality over the interpretations of the arguments and refers to equality of the referred-to parties, not the terms:

$$\mathcal{I} \models (t_1 = t_2) \text{ iff } (t_1^{\mathcal{I}} =_{\text{pty}} t_2^{\mathcal{I}}) .$$

The relation $=_{\text{pty}}$ is well defined and evaluatable for the interpretation domain $\mathfrak{D}_{\text{pty}}$ of the type. The interpretation of a term of type **pty** is a system-wide unique identifier for the party that need not be further specified. The interesting aspect for this equality relation is its dependence on the system state specified through the interpretation \mathcal{I} .

The $=$ -relation on *subject*, *holder*, *certifier*, and *recipient* attributes refers to the equality of parties, and not equality of the terms used for referring to them. That is, even for different terms that refer to the same party, equality can hold, that is, for different pseudonyms referring to the same party. It is important to mention that this predicate cannot be evaluated by every party which exactly reflects the power of pseudonymous interactions: The holder of pseudonyms can link different pseudonyms to belong to the same party through the terms it uses, the recipient cannot do so by default, unless given additional information by the holder, thus, those equality relationships remain hidden at the syntactic level. This shows the appropriateness of our choice of using a logic with equality for our modeling purposes.

Attribute names. We do not need to express predicates over **att**-typed arguments as such arguments are only used as arguments of functions corresponding to identifier objects, identities, opaque identities, and conditionally released identities.

4.11.4 Interpretation for a System State

We next define an interpretation \mathcal{I} which determines the meaning of formulae in a concrete system in a given system state \mathcal{G}^δ . This formally connects the language fragments of \mathcal{L}^p , e.g., $\mathcal{L}^{p/c}$, corresponding to specific technologies, to the cryptographic protocols that have been executed until \mathcal{G}^δ has been reached. We note that we are interested in a single relevant interpretation for a system at state \mathcal{G}^δ only, which is a function of \mathcal{G}^δ , or, equivalently, the protocols executed until then.

Considering the discussions above, the interpretation function \mathcal{I} is a surjective function from the (state-dependent) types to their (state-dependent) interpretation domains. The function corresponds to a partition shown in (4.25) comprising the functions of the different types.

$$\begin{aligned} \mathcal{I} = & \mathcal{I}^{\text{idfo}} \cup \mathcal{I}^{\text{id}} \cup \mathcal{I}^{\text{oid}} \cup \mathcal{I}^{\text{crid}} \cup \mathcal{I}^{\text{att}} \cup \mathcal{I}^{\text{idf}} \cup \mathcal{I}^{\text{pty}} \cup \\ & \mathcal{I}^{\text{int}} \cup \mathcal{I}^{\text{date}} \cup \left(\bigcup_{\xi \in \{Y, M, D, h, m, s\}} \mathcal{I}^{\text{date}^\xi} \right) \cup \mathcal{I}^{\text{str}} \cup \mathcal{I}^{\text{bool}} \end{aligned} \quad (4.25)$$

When we want to refer to the interpretation $\mathcal{I}^{\tilde{\tau}}$ for a type $\tilde{\tau}$ in a specific system state \mathcal{G}^δ , we may make this explicit with the notation $\mathcal{I}^{\tilde{\tau},\delta}$.

Every execution of a cryptographic protocol, the functionality of which in terms of the data representation is discussed in Chapter 5, induces a state transition from \mathcal{G}^δ to $\mathcal{G}^{\delta+1}$. This state transition induces a change of the state-dependent types and corresponding interpretation domains as well as the interpretation function. The execution of one instance of a protocol is assumed to be atomic. We consider a system state \mathcal{G}^δ to be equivalent to the corresponding sequence of protocol instances executed for reaching it.

We anticipate in the following discussion partly how the execution of each of the cryptographic protocols discussed in detail in terms of data modeling in Chapter 5 influences the system state. Particularly, we also use some terminology and symbols introduced in Chapter 5 without being introduced now. For a full understanding of the following discussion, Chapter 5 is a prerequisite.

4.11.4.1 Establishing a Registration Subject Identifier

An instance of the protocol `EstIdtfRelReg` between parties **A** and **B**, with an optional delegater **D**, creates a new registration identifier object referred to through the term \mathbf{t}'_p , modeled through its corresponding function $\mathcal{X}^{\text{idf}}_{\mathbf{t}'_p}$. Execution of the protocol creates a new term for referring to **A** or new terms for **A** and **D** in case of delegation, reflected in an updated interpretation function

$$\mathcal{I}^{\text{pty},\delta} := \mathcal{I}^{\text{pty},\delta-1} \cup \mathcal{I}^{\text{pty},\Delta_{\mathbf{t}'_p}},$$

where $\mathcal{I}^{\text{pty},\Delta_{\mathbf{t}'_p}} = \{(\mathbf{t}'_{h,\mathbf{A}}, \mathbf{A})\}$ in the case of no delegation and $\mathcal{I}^{\text{pty},\Delta_{\mathbf{t}'_p}} = \{(\mathbf{t}'_{h,\mathbf{B}}, \mathbf{A}), (\mathbf{t}'_{s,\mathbf{A}}, \mathbf{D})\}$ in the case of delegation. Recall that the term renaming function for \mathbf{t}'_p is defined as $\varrho^{\text{idf}}_{\mathbf{t}'_p} = \{\}$ in the non-delegation case and $\varrho^{\text{idf}}_{\mathbf{t}'_p} = \{(\mathbf{t}_{h,\mathbf{A}}, \mathbf{t}'_{h,\mathbf{B}})\}$ in the delegation case. Execution of the protocol updates the interpretation function

$$\mathcal{I}^{\text{idfo},\delta} := \mathcal{I}^{\text{idfo},\delta-1} \cup \mathcal{I}^{\text{idfo},\Delta_{\mathbf{t}'_p}},$$

with $\mathcal{I}^{\text{idfo},\Delta_{\mathbf{t}'_p}} = (\mathbf{t}'_p, \mathcal{X}^{\text{idf}}_{\mathbf{t}'_p})$, where $\mathcal{X}^{\text{idf}}_{\mathbf{t}'_p} = \{(\text{holder}, \mathbf{A}), (\text{subject}, \mathbf{A}), (\text{subjectId}, \mathbf{t}'_{\text{sid},\mathbf{A}})\}$ in the non-delegation case and $\mathcal{X}^{\text{idf}}_{\mathbf{t}'_p} = \{(\text{holder}, \mathbf{A}), (\text{subject}, \mathbf{D}), (\text{subjectId}, \mathbf{t}'_{\text{sid},\mathbf{A}})\}$ in the delegation case, and optionally comprising the tuple $(\text{domain}, \mathbf{t}_{\text{dm}})$ in case of a domain identifier object.

4.11.4.2 Establishing a Subject Identifier

A protocol instance `EstIdtfRel` between parties **A** and **B**, with an optional delegater **D**, creates a new identifier object \mathbf{t}'_p , modeled through its corresponding function $\mathcal{X}^{\text{idf}}_{\mathbf{t}'_p}$. Execution of the protocol creates a new term for **B** referring to **A** and additionally a new term for referring to **D** in case of delegation:

$$\mathcal{I}^{\text{pty},\delta} := \mathcal{I}^{\text{pty},\delta-1} \cup \mathcal{I}^{\text{pty},\Delta_{\mathbf{t}'_p}}$$

where $\mathcal{I}^{\text{pty}, \Delta_{t'_p}} = \{(t'_{h, \mathbf{B}}, \mathbf{A})\}$ for the non-delegation case and $\mathcal{I}^{\text{pty}, \Delta_{t'_p}} = \{(t'_{h, \mathbf{B}}, \mathbf{A}), (t'_{s, \mathbf{B}}, \mathbf{D})\}$ in the delegation case. Additionally, execution of the protocol updates the interpretation function

$$\mathcal{I}^{\text{idfo}, \delta} := \mathcal{I}^{\text{idfo}, \delta-1} \cup \mathcal{I}^{\text{idfo}, \Delta_{t'_p}}.$$

where $\mathcal{I}^{\text{idfo}, \Delta_{t'_p}} = (t'_p, \mathcal{X}_{t'_p}^{\text{idf}})$ and $\mathcal{X}_{t'_p}^{\text{idf}} = \{(holder, \mathbf{A}), (subject, \mathbf{A}), (subjectId, t'_{sid, \mathbf{A}})\}$ in case of no delegation and $\mathcal{X}_{t'_p}^{\text{idf}} = \{(holder, \mathbf{A}), (subject, \mathbf{D}), (subjectId, t'_{sid, \mathbf{A}})\}$ in the delegation case, always with an optional domain $(domain, t_{dm})$.

4.11.4.3 Establishing an Identity Relationship

A protocol instance of **EstIdtyRel** between parties **A** and **B**, with an optional delegater **D**, creates a new identity relationship between **A** and **B**, either with **A** or **D** as subject, in case of it being without or with delegation, respectively. The term t'_c used by both **A** and **B** for referring to the created identity corresponds to a function $\mathcal{X}_{t'_c}^{\text{id}_y}$ which is the interpretation for the identity, resulting in

$$\mathcal{I}^{\text{id}, \delta} := \mathcal{I}^{\text{id}, \delta-1} \cup \{(t'_c, \mathcal{X}_{t'_c}^{\text{id}_y})\}.$$

Thereby, $\mathcal{X}_{t'_c}^{\text{id}_y}$ is the function representing the identity as discussed in detail in other parts of this work. No new terms for referring to the parties are created during this protocol, thus

$$\mathcal{I}^{\text{pty}, \delta} := \mathcal{I}^{\text{pty}, \delta-1}.$$

Regardless of executing an instance of **EstIdtyRel**, each pair of identity type and certifier for which identities are to be issued by the certifier gives rise to a dummy identity referred to through c and modeled through $\mathcal{X}_c^{\text{id}_y}$, thus, giving rise to the tuple $(c, \mathcal{X}_c^{\text{id}_y})$ being added to $\mathcal{I}^{\text{id}, \delta-1}$, where \mathcal{G}^δ is the system state corresponding to the creation of the identity type by the certifier.

4.11.4.4 Releasing Data

An instance of the protocol **RelData** between parties **A** and **B**, with an optional delegater **D** being referred to, releases a data statement to **B**. Thereby, only existing terms for referring to the parties are used, that is, the interpretation function for parties remains the same:⁴⁰

$$\mathcal{I}^{\text{pty}, \delta} := \mathcal{I}^{\text{pty}, \delta-1}.$$

New terms are created for referring to identities, as well as for to-be-established conditionally released identities and opaque identities, and lead to updates of the corresponding interpretation functions for those types of objects.

⁴⁰Note, though, that in a practical implementation, this protocol can lead to the implicit establishment of new identifier relationships as part of the protocol for reasons of efficiency and more concise protocol specification.

Let $\mathbf{t}_1^{\text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\text{id}}$ be the identities corresponding to credential-based identity relationships of the party and dummy identities referred to in the formula to be released. Execution of the protocol extends \mathcal{I} through introducing new terms $\mathbf{t}_1^{\prime \text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\prime \text{id}}$ for identities:

$$\mathcal{I}^{\text{id}, \delta} := \mathcal{I}^{\text{id}, \delta-1} \cup \bigcup_{i=1}^{\ell^{\text{id}}} (\mathbf{t}_i^{\prime \text{id}}, \mathcal{X}_{\mathbf{t}_i^{\text{id}}}^{\text{id}}).$$

Let $\mathbf{t}_{\hat{\ell}^{\text{crid}}+1}^{\text{crid}}, \dots, \mathbf{t}_{\ell^{\text{crid}}}^{\text{crid}}$ be the terms for referring to the conditionally released identities generated in the protocol and $\mathbf{t}_{\hat{\ell}^{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ the terms for referring to the generated opaque identities. The interpretation function is extended as presented next, using new terms for the newly created conditionally released identities and opaque identities and adding them to the corresponding functions:

$$\begin{aligned} \mathcal{I}^{\text{crid}, \delta} &:= \mathcal{I}^{\text{crid}, \delta-1} \cup \bigcup_{i=\hat{\ell}^{\text{crid}}+1}^{\ell^{\text{crid}}} (\mathbf{t}_i^{\text{crid}}, \mathcal{X}_{\mathbf{t}_i^{\text{crid}}}^{\text{crid}}) \quad \text{and} \\ \mathcal{I}^{\text{oid}, \delta} &:= \mathcal{I}^{\text{oid}, \delta-1} \cup \bigcup_{i=\hat{\ell}^{\text{oid}}+1}^{\ell^{\text{oid}}} (\mathbf{t}_i^{\text{oid}}, \mathcal{X}_{\mathbf{t}_i^{\text{oid}}}^{\text{oid}}). \end{aligned}$$

For a formula to be released, the context of whom the formula is to be released to influences the applicable terms for the holder and subject of the parties the formula makes statements about. The constraints on terms \mathbf{t}_h and \mathbf{t}_s referring to a holder and a subject are that a term \mathbf{t}_h may only be a term that **A** has previously established with **B** about party **A** and, analogously, for the delegator **D**, a term referring to it as subject may only be a term established by **A** with **B** about **D** previously, through an instance of a protocol for establishing an identifier object. In case those constraints are not met, the formula may still be true according to the denotational semantics, though a cryptographic proof of it that might be accepted by **B** cannot be verified any more by the data recipient **B**.

Following the above, the system state \mathcal{G}^δ gives rise to the following state-dependent type definitions and interpretation domains for the non-self-referential types as defined in (4.26), thereby defining the permissible terms in formulae and their interpretations, respectively, in a given system state.

$$\begin{aligned} \text{idfo}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{idfo}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{idfo}} &= \text{img}(\mathcal{I}^{\text{idfo}, \mathcal{G}^\delta}) \\ \text{id}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{id}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{id}} &= \text{img}(\mathcal{I}^{\text{id}, \mathcal{G}^\delta}) \\ \text{crid}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{crid}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{crid}} &= \text{img}(\mathcal{I}^{\text{crid}, \mathcal{G}^\delta}) \\ \text{oid}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{oid}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{oid}} &= \text{img}(\mathcal{I}^{\text{oid}, \mathcal{G}^\delta}) \\ \text{pty}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{pty}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{pty}} &= \text{img}(\mathcal{I}^{\text{pty}, \mathcal{G}^\delta}) \\ \text{idf}^{\mathcal{G}^\delta} &= \text{preimg}(\mathcal{I}^{\text{idf}, \mathcal{G}^\delta}) & \mathfrak{D}_{\text{idf}} &= \text{img}(\mathcal{I}^{\text{idf}, \mathcal{G}^\delta}) \end{aligned} \tag{4.26}$$

4.11.5 Extension: Operators on Attributes

Our extension of Sec. 4.4.17 of introducing operators for attributes requires further consideration and makes the definitions of the predicates and the underlying logic more non-standard. Recall that an operator allows for an inequality to be expressed on an attribute in the definition of the object, e.g., identity, without using a formula. Next, we sketch the basic ideas for such extension without giving full details.

For this extension, a function $\mathcal{X}_c^{\text{id}}$ modeling an identity and the functions corresponding to the other kinds of objects are redefined such that they map an attribute name not only to the typed attribute value, rather they map an attribute name to a tuple $(d :: \beta, \text{op} :: \text{Op})$ comprising the value d of type β of the attribute and an operator op . That is, the functions do not map any more to an element of a basic data type alone, which changes the definition of predicates when an argument is an attribute reference.

Interpretations are done analogously by mapping identities and other cryptographic objects, represented as function symbols, to the actual functions of the interpretation domain of the types. Considering the definitions given earlier, an attribute reference is interpreted with a pair $(d :: \beta, \text{op} :: \text{Op})$. Constants and variables denoting elements of basic data types are typed with such. This leads to the situation that predicates on the basic data types need to take the definition of attribute references through operator-attribute tuples and terms typed with basic types into consideration.

Concretely, this leads to a substantial complication of the definition of the predicates on the basic types and the recursive definition of terms recursively composed through operators, comprising also expressions. The semantics of those predicates is defined through reducing them to the predicates expressed without operators and accounting for the operator-value tuples through distinguishing cases for the operators. Recursively composed terms are handled similarly by considering cases depending on the operators of the components.

Using operators complicates the semantics and the deductive system of the logic substantially by deviating from the standard definitions of the data types and making the relation predicates substantially more involved, as well as the interpretations. This is the main reason why operators are maintained only as an optional extension to our system.

Predicates. Next, we define the semantics for the predicate $\text{Eq}_{(\text{int} \times \text{Op}_{\text{int}}) \times \text{int}}$ on an integer-operator argument and an integer-only argument to give an example for the semantics specification when having operator support.

$$\mathcal{I} \models \text{Eq}_{(\text{int} \times \text{Op}_{\text{int}}) \times \text{int}}((d_1, \text{op}_1), d_2) \quad \text{iff} \quad d_1^{\mathcal{I}} =_{\text{int}} d_2^{\mathcal{I}} \text{ and } \text{op}_1 = \text{eq}$$

The other required predicates can be specified using similar ideas, though are not presented here.

Expressions on integers. Recursive definition of terms also needs to take the operators into account. Expressions on the int -type are a good example for this and

we sketch the approach for such expressions as one particular example of recursive definition of terms.

Let an expression exp be defined as is standard as a sum of products and a constant term. Then, the interpretation $\hat{e}^{\mathcal{I}}$ of the value \hat{e} , without operator, of the expression exp is defined as the expression on the interpreted values with operators of the type int , where $\pi^i(t)$ is the i -th projection of tuple t :

$$\begin{aligned} \hat{e}^{\mathcal{I}} := & u_{j_1}^{\mathcal{I}} \cdot \pi^1((o_{j_1}.a_{k_1})^{\mathcal{I}}) + \dots + u_{j_i}^{\mathcal{I}} \cdot \pi^1((o_{j_i}.a_{k_i})^{\mathcal{I}}) + \dots + \\ & u_{j_l}^{\mathcal{I}} \cdot \pi^1((o_{j_l}.a_{k_l})^{\mathcal{I}}) + u_{j_{l+1}}^{\mathcal{I}}. \end{aligned}$$

The operator of the expression is derived as shown in (4.27), where the expression is undefined in case the last of the cases applies. Let for the operator derivation below op_{j_i} be defined as $\pi^2((o_{j_i}.a_{k_i})^{\mathcal{I}})$, where variable i ranges from 1 to l and let \perp represent a value that is undefined.

$$exp^{\mathcal{I}} := \begin{cases} (\hat{e}, \text{eq}) & \text{if } \forall i : \text{op}_{j_i} = \text{eq} \\ (\hat{e}, \text{lt}) & \text{if } \exists i : \text{op}_{j_i} = \text{lt} \wedge \forall i' \neq i : \text{op}_{j_{i'}} = \text{eq} \vee \text{op}_{j_{i'}} = \text{leq} \vee \text{op}_{j_{i'}} = \text{lt} \\ (\hat{e}, \text{leq}) & \text{if } \exists i : \text{op}_{j_i} = \text{leq} \wedge \forall i' \neq i : \text{op}_{j_{i'}} = \text{eq} \vee \text{op}_{j_{i'}} = \text{leq} \\ (\hat{e}, \text{geq}) & \text{if } \exists i : \text{op}_{j_i} = \text{geq} \wedge \forall i' \neq i : \text{op}_{j_{i'}} = \text{eq} \vee \text{op}_{j_{i'}} = \text{geq} \\ (\hat{e}, \text{gt}) & \text{if } \exists i : \text{op}_{j_i} = \text{gt} \wedge \forall i' \neq i : \text{op}_{j_{i'}} = \text{eq} \vee \text{op}_{j_{i'}} = \text{geq} \vee \text{op}_{j_{i'}} = \text{gt} \\ \perp = & \text{otherwise} \end{cases} \quad (4.27)$$

An observation is that not all combinations of operators in the arguments lead to valid expressions which substantially constrains how expressions can be expressed. Practically, this is not a notable limitation when considering the use cases for operators.

The by far most relevant case is the one of all operators being **eq** which is the case of standard expressions on integers. The other cases capture requirements of rather specific use cases which compute expressions over integers with operators and have multiple restrictions on the integers to be considered. This functionality of computing expressions on attributes with operators other than the equality operator can be considered optional due to its specificity.

4.11.6 Discussion

The data model and its semantics are constructed such that the semantics applies throughout all formulae in a system at a given state, not only a single formula or set of formulae in isolation. Concretely, this means that terms have global validity in the system, not only in a limited context, such as an authentication transaction. This partially leads to the complications in our data model, e.g., the (re)naming of terms to refer to entities.

A substantial advantage of a semantics with those properties is that a recipient party of data statements can derive new formulae from multiple received formulae

and the semantics equally applies to such formulae as well. For example, this is relevant for supporting incremental authentication. For different approaches where a semantics is defined for a single formula viewed in isolation, the semantics does not apply for formulae derived through the logical calculus from multiple such formulae.

Capturing aspects such as revocation of identities or consumption of identities through their usage, that is, temporal aspects or aspects of non-monotonicity would require consideration of non-standard logic, e.g., linear logic, for expressing this. We have modeled such aspects as operational semantics of the underlying protocols. Keeping those aspects outside of our data model and semantics and constraining it to capturing basic identity data aspects makes it compatible with the monotonicity property of first-order logic and allows for interpreting a formula in the latest system state $\mathcal{G}^{\delta_{\max}}$ where the formula has been derived from formulae released in earlier system states.

Our semantics models the actual relations that hold in terms of identity management in a system at a given state, e.g., the attribute values of identities, or holders, subjects, and certifiers of the identities in the system, rather than being constrained to modeling what can be (cryptographically) proven by parties to other parties based on their portfolio and a concrete suite of mechanisms. That is, a formula true under an interpretation corresponding to a system state may be not provable using certain cryptographic protocols, because mechanisms constrain which formulae can be proven. This is intended rather than being a limitation, because the primary purpose of a semantics is defining the denotation of each formula in a system. Cryptographic or other techniques for establishing the correctness of a released formula, e.g., a proof of equality of the subject of two identities, govern which statements can be established by a party using cryptographic material it holds.

Related to this, there is a difference between the statements that can be proven by a party through cryptographic mechanisms and the statements that can be syntactically deduced from those statements in our calculus, while both are statement-type formulae and captured by our semantics.

Also, there exists a gap between formulae that are fulfilled according to the semantics of the language $\mathcal{L}^{p/g}$ and what can be derived by a party having received the formulae. Less can be derived, e.g., linking different pseudonyms to the same holder or subject is possible only if the equality of those has been established by the sender. This precisely captures the idea of privacy-enhancing protocols and has led to the complications with the term for referring to objects and semantics.

For a clean modeling of formulae comprising disjunctions through the semantics, we have used the approach of introducing what we refer to as *dummy identities*. For those parts of a formula which are not fulfilled through identities or other objects related to a party, those globally defined dummy identities are used for being able to interpret such a formula. This is crucial for the semantics of a data-minimizing formula a party wants to release, which refers both to credentials and pseudonyms the party holds as well as credentials that it does not hold. This has solved only one of the multiple challenges related to disjunctions in formulae.

Chapter 5

Cryptographic Protocol Integration

In the current chapter we discuss the application of the data representation concepts of Sec. 4.4 for expressing data in our system. This comprises the representation of formulae corresponding to identifier objects and identities and formulae for the specification of the interfaces of the cryptographic protocols to be executed to establish those, as well as for the data release protocol to utilize those objects for releasing data. We present the different languages required for expressing those system aspects and discuss how they relate to each other. Also, we discuss how the protocols relate to the data stored by a party and how output of a protocol is handled by a party. Because the presented protocols are a means for realizing the channel transformation rules presented in our channel model of Chapter 3, we explain how the protocols relate to and thereby realize the authentication functionality of this model. Our holistic view over all protocols a party can execute, instead of only the data release protocol as in many related works, is, by formally relating the different protocols, an important contribution.

For our presentation, we go into a deep level of detail on the data representation beyond the conceptual level for avoiding ambiguities and facilitating an implementation of the overall system. Thereby, the presentation touches on crucial aspects related to an implementation of the system, such as interfaces for the protocols.

The protocol interface discussions can be seen as a substantial extension and generalization of earlier work by the author [CSZ06b, CSZ06a] which had interfaces of the protocols for a privacy-enhancing PKI system as one of its contributions. The current work is based on a generalization of the protocols as well as formally linking them to the data representation. An implementation architecture of further aspects of the protocol implementation is presented in work by the author [Som11].

As part of our presentation of the protocol interfaces, we particularly show how cryptographic objects are created in protocols, stored in the party's portfolio and repository, and re-used between instances of (different) protocols by being referred to through terms of the logic-based data representation across those protocol instances. Preventing undesired linkabilities being established between protocol instances through the data representation and obtaining a data representation which is capable of utilizing the reasoning capabilities of our logic, e.g., for matching a party's identifier and identity relationships against a data request, require the approach being explained for creating and renaming constant terms in our logic-based language used to refer to objects or parties. This handling of terms introduces additional complexity in contrast to comparable languages which may seem unproportionate at a first glance. Though, this term handling provides substantial benefits, such as the full support of automated deductions in different contexts, particularly for matching a party's repository formulae against data requests, authorization decisions, and making deductions in our logic over multiple received data statements of a party, a semantics valid over all data statements in a system and not only defined for individual formulae, and full integration of delegation support into the logic-based modeling and reasoning.

In this chapter we focus on protocols based on credential systems as underlying cryptographic technology. See Chapter 6 for a summary of our approach for the runtime generation of cryptographic protocols for releasing data. A generic, protocol-independent, view of our system for the support of further certification technologies besides credential systems is presented in earlier work [Som11]. Particularly, we show how cryptographic approaches are used for realizing registration domains and delegation registration domains.

We note that the notation we use for addressing entities such as parties or identifier objects in the following discussions diverges from the notation used for the channel model in Chapter 3. The reason is that in the discussions below, a single protocol may realize different aspects of a rule in the model and thereby we cannot reuse the same notation, e.g., different certifiers C_i and C_{k+1} in rule **AuthDele-Cred** are represented with **B** in the protocol for establishing an identity relationship. Related to the protocols for establishing identifier relationships and identity relationships, we interchangeably use the terminology of establishing the corresponding identifier object or identity.

5.1 Prerequisites

We next present some prerequisites needed for a discussion of the protocols.

5.1.1 Cryptographic Object Structures

For each object type, that is, each identity type, conditionally released identity type, and opaque identity type, a corresponding *structure*, or *cryptographic object structure*, determining the technical encoding for the cryptographic object is required in order to be able to compute cryptographic protocols based on those. A structure is defined for each certificate type issued by a given certifier, opaque identity type, or conditionally released identity type either explicitly or implicitly.

Recall that the type of an object defines its attributes and their data types. While identity types are specified explicitly, the type of an opaque or conditionally released identity is given implicitly through the relation of the attributes of the object with attributes of other objects through relation predicates in the formula through a data release protocol of which the cryptographic object gets created, made explicit during the protocol, and stored with the created cryptographic object. The reason for not specifying the types of opaque or conditionally released identities explicitly upfront is to not complicate policy authoring.

The structure for an object type determines the technical encoding for the representation of the data expressed in the object specification formula, or, equivalently, expressed in the corresponding formula in the semantics. This encoding does not have any meaning at the level of the data model, but rather is necessary for the technical implementation of the semantics expressed through the formula in the data model using cryptographic techniques.

The main part of the encoding specification of a structure is a function \mathcal{M} specifying the mapping from attributes specified in the object specification formula to integer attributes of the corresponding cryptographic object. The preimage of the function comprises the attributes of the object and the image contains triples each comprising the data type of the attribute, a mapping algorithm, and a tuple of indices. The data type and mapping algorithm determine how the attribute value is mapped to its cryptographic representation, the index tuple comprises the indices of attributes of the cryptographic object that the attribute is mapped to.

For identities, the structure is denoted *certificate structure* as it specifies the technical encoding for the private certificate used for realizing the identity. A certificate structure for a given certificate type by a given certifier is specified by the certifier or another designated party. Integrity of a certificate structure is ensured through certification using traditional PKI approaches or a Fiat-Shamir signature-based PKI in the case of pseudonymous certifiers. Thus, for an identity, the structure needs to be obtained by a party that intends to execute a cryptographic protocol referring to an identity of this type and certifier from the certifier of the identity.

For a certificate structure, the encoding is specified through the function \mathcal{M} and the supported features and details regarding their cryptographic realization through integer attributes of the certificate through the function \mathcal{F} .

For conditionally released and opaque identities, the structure is determined by the party in the instance of the RelData protocol in which it is created and stored as metadata expressed over the cryptographic objects. As the latter structures are determined through the formula being released and proven, using the same approach

of PKI-based integrity protection is not a flexible approach.

5.1.2 Dummy Identities and Certificates

For realizing cryptographic proofs comprising disjunctions over predicates expressed over identities that the prover cannot fulfill with the identities of its identity relationships, we make use of the concept of *dummy identities* introduced with the semantics in Def. 4.3 in Sec. 4.11.

For each dummy identity c , an object specification formula ϕ_c describes its attributes analogous to the identity specification formula or formulae specifying an identity in an identity relationship. The difference is, though, that only statements about the type and certifier are made—all other attributes are left unspecified instead of being populated with arbitrary values in order to avoid problems with policy matching.

The constant term for referring to a dummy identity is computed by applying a collision-resistant one-way function to a canonicalized representation of the attribute values of the identity concatenated with the corresponding certifier public key. This avoids, with overwhelming probability, that different dummy identities for the same type and certifier can have the same term for referring to them. The term construction is analogous to the derivation of terms for identities from cryptographic material realizing identities, and so is the purpose.

A dummy identity has corresponding certifier specification formulae specifying an identifier object and identity that have the certifier as subject. The integrity of those formulae can be verified cryptographically by the party through PKI using standard mechanisms or a Fiat-Shamir based signature in case of a pseudonymous certifier.

Dummy identities are utilized for finding satisfying formulae for data requests comprising disjunctions by using automated deduction in our logic calculus. For such requests, a dummy identity can match with predicates in an OR-branch of a request for which the party does not have identity relationships, that is, credentials, consistent with those predicates. Through its definition, a dummy identity always matches correctly the type and certifier predicates anywhere in the formula, while predicates in OR-branches referring to the dummy identity need not hold when other OR-branches are fulfilled.

A dummy identity with its object specification formula can be created by the certifier of identities of such types and its integrity protected through a signature with the issuing key of the certifier. Architecturally, specification formulae of dummy identities are conveyed to parties as part of ontologies as explained in Sec. 4.10. Using such ontology, a party can find a dummy identity as a matching identity for predicates in an unfulfilled OR-branch of a data request.

A dummy identity referred to as t_i^{idy} is associated with a *dummy certificate* χ_i^{crt} comprising only a certificate structure and the attribute tuples, though no cryptographic certificate, that is, signature. The certificate structure is required for constructing cryptographic proofs in which proofs for assertions about the dummy identity are to be simulated using zero-knowledge proof of knowledge simulation

techniques as part of a disjunctive statement. The certificate structure is the exact same certificate structure the certifier provides for any certificate of the same type. A dummy identity is also related to the public key of the certifier—this key is used by all parties executing protocols based on private certificates of this type issued by the certifier. A dummy certificate for a dummy identity can be constructed by a party based on the certificate structure and the specification formula for the identity.

5.2 Establishing Identifier Relationships

The concept of identifier relationships has been introduced in Sec. 4.7.1. The below-presented protocols for establishing identifier relationships are suitable for the case of party **B** being known under a (public) pseudonym to **A**. The two-way pseudonymous case where the other party does not reveal a pseudonym upfront can be handled in a similar way, where the difference is that initially **B** is not known by a pseudonym which needs to be reflected in the protocols and their interfaces.

5.2.1 Protocol for Establishing a Registration Identifier Relationship

The protocol `EstldtfRelReg` is used by a party **A** to create a new *identifier object* for itself or another party **D** as subject with a party **B**, in the context of a registration of **A** in the sense of our model of Chapter 3 or registration with a delegation certifier. In the further case, a *registration identifier object* is created, in the latter case, a *delegation registration identifier object*. The identifier object realizes a *pseudonym* and is technically represented at both involved parties **A** and **B** through a new *identifier relationship*. In our channel model, this protocol is executed whenever a party intends to establish a pseudonymous relationship with a registration certifier or with a certifier for the purpose of establishing a delegation relationship. In the idealized model of a pseudonym and credential system of Lysyanskaya et al. [LRSW00] and Camenisch and Lysyanskaya [CL01], a registration is done once for each user only, thereby imposing constraining assumptions. We next present the inputs and outputs of the parties **A** and **B** for an instance of the protocol.

`EstldtfRelReg` protocol interface:

Input	
A	$\phi_A, \{\diamond(\text{"estWith"}, t_{sid_B}), \diamond(\text{"subj"}, t_s)\}$
B	—
Output	
A	$\phi'_A, \{\diamond(\text{"role"}, \text{"holder"}), \diamond(\text{"estWith"}, t_{sid_B}),$ $\diamond(\text{"pubCryptoMaterial"}, \widehat{\kappa}^{gc}), \diamond(\text{"privCryptoMaterial"}, \widetilde{\kappa}^{gc}),$ $\diamond(\text{"renFct"}, \varrho^{idf})\}, success_A$
B	$\phi'_B, \{\diamond(\text{"role"}, \text{"recipient"}), \diamond(\text{"estWith"}, t_{sid_B}),$ $\diamond(\text{"pubCryptoMaterial"}, \widehat{\kappa}^{gc})\}, success_B$

Notational concepts. For the interface for the protocol `EstIdtfRelReg`, we introduce the general approach we employ for specifying interfaces of protocols of using, for the input and output of each of the involved parties, *formulae* for making statements about objects and using *metadata* predicates, e.g., for expressing pseudonym identifiers and cryptographic material.

Formulae. The formulae are based on the concepts of the data model of Sec. 4.4. Different constrained languages are used for representing those formulae. Free variables in a formula are used for indicating that those be instantiated with concrete terms in a corresponding output formula.

Metadata. The metadata predicates are implicitly associated with the formula through the interface notation and expressed through \diamond -predicates. A metadata predicate can be of one of the following kinds: First, the predicate comprises a label χ^\diamond specifying the kind of metadata and a metadata item ξ^\diamond as in (5.1). Second, it—in addition—comprises a term o^\diamond for referring to the object that the metadata applies to as in (5.2). The object needs to be one that is referred to in the formula that the metadata predicate is associated with. Through this approach we can associate metadata with formulae in general and also with specific objects referred to therein. Note that, by referring to objects in metadata predicates using terms of our data representation approach, we introduce parts of the formalized data representation based on the data model into the representation of metadata. Metadata predicates being part of the protocol output can be stored transparently with an established identifier relationship or identity relationship or other entry.

$$\diamond(\chi^\diamond, \xi^\diamond) \tag{5.1}$$

$$\diamond(o^\diamond, \chi^\diamond, \xi^\diamond) \tag{5.2}$$

Explicit and implicit input. The interface of a protocol as we specify it comprises the input and output elements of the protocol, though, there are different

ways of how elements can be provided as input. On the one hand, an item can be explicitly passed as part of the protocol input at the API level of the implementation of the protocol, on the other hand an item can be retrieved by the protocol's implementation from the party's data repositories or online services based on further information available in the input, particularly the terms referred to in input formulae. The latter approach to protocol input is denoted by putting the respective input element in between $\langle\langle \rangle\rangle$ symbols. We chose this notational approach in order to make all required inputs and outputs for the protocol explicit in the interface, while constraining the input elements required to be explicitly passed in an implementation to the minimum possible and therefore following the architecture paradigm of encapsulating the complexity of a subsystem within it. This discussion is already very close to an implementation and corresponding architecture—see Sommer [Som11] on related aspects of an implementation architecture—one goal of it being to keep the protocol interface of the credential protocols reasonably simple.

Optional interface arguments. An optional interface argument is indicated as being optional by including it between $[]$ symbols.

Omitted interface elements. Further cryptographic material such as what we refer to as *system parameters*, see, e.g., a related protocol specification [Sec10], is required for each of the protocols for realizing a pseudonym and credential system, though, is not mentioned in the specification of the protocol interfaces.

Also, further not mentioned metadata for internal bookkeeping and, e.g., data tracking functionality, such as a timestamp of protocol execution, may be required for a practical system implementation at both parties for all protocols. We do not make any of those further elements explicit in order to not clutter the notation more than necessary.

5.2.1.1 Input

Formulae. We use the approach of data modeling of Chapter 4.4 for representing the formulae, $\phi_{\mathbf{A}}$ for the input of party \mathbf{A} , $\phi'_{\mathbf{A}}$ for \mathbf{A} 's output, and $\phi'_{\mathbf{B}}$ for \mathbf{B} 's output as shown in (5.3). The output formulae are derived from the input formula through the protocol. The free variables in the input $\phi_{\mathbf{A}}$ are replaced with constant terms in the output after a successful protocol execution. In an implementation, \mathbf{A} 's input $\phi_{\mathbf{A}}$ can be derived from a *template* at party \mathbf{A} , similar to how a certifier can use templates for the formulae for determining identity relationships to be established.

$$\begin{aligned}\phi_{\mathbf{A}} &= T_p.\text{holder} = t_{h,\mathbf{A}} \wedge T_p.\text{subject} = t_{s,\mathbf{A}} \wedge \text{Eq}(T_p.\text{subjectId}, t_{sid,\mathbf{A}}) \\ \phi'_{\mathbf{A}} &= t'_{p,\text{holder}} = t'_{h,\mathbf{A}} \wedge t'_{p,\text{subject}} = t'_{s,\mathbf{A}} \wedge \text{Eq}(t'_{p,\text{subjectId}}, t'_{sid,\mathbf{A}}) \\ \phi'_{\mathbf{B}} &= t'_{p,\text{holder}} = t'_{h,\mathbf{B}} \wedge t'_{p,\text{subject}} = t'_{s,\mathbf{B}} \wedge \text{Eq}(t'_{p,\text{subjectId}}, t'_{sid,\mathbf{B}})\end{aligned}\tag{5.3}$$

For the input formula $\phi_{\mathbf{A}}$ in (5.3), the identifier object to be created is represented through a free variable T_p , with the variable expressing the meaning that an

identifier object will be created in the protocol. The holder of the object is indicated through a constant or variable term $t_{h,A}$ referring to the party itself. The subject of the identifier object is specified by the attribute *subject* through term $t_{s,A}$, being a constant or variable for A itself or a third party D . The protocol being used for establishing the identifier relationship is indicated through the *protocolSuite* attribute like for an identity. We may omit this attribute in the specified formulae in the text for the reason of brevity.

Whether the *holder* and *subject* are constants or variables depends on whether the pseudonym is to be established for A itself or a third party, resulting in the following cases: Case (1) of establishing a registration identifier object: The identifier object is to be created for A itself, thus both attribute values are represented through the same variable, that is, $t_{h,A} = t_{s,A}$, because holder and subject are the same party and there is no constant available yet to refer to it. The meaning is that a new term referring to A is created in the protocol associated to both holder and subject of the identifier object. Case (2) of establishing a delegation registration identifier object: For the case of the identifier object to be established about a third party D , that is, the subject term referring to D , the holder is a constant term $t_{h,A}$ referring to A and the subject a variable $t_{s,A}$ referring to D that gets instantiated through the protocol. The term for party A is the term used by the party to refer to itself locally throughout the registration domain this delegation registration identifier object is to be created in and has been created for one registration identifier object—of possibly multiple ones—with A being the subject.

Metadata. The metadata predicate $\Diamond(\text{“estWith”}, t_{sid_B})$ specifies the pseudonym identifier of the party with which the identifier is to be established, where $t_{sid_B} :: \text{idf}$. This identifier is available in all practical settings where B acts under a public pseudonym. In case its pseudonym with A has not been established yet, the protocol interface needs to be able to reflect this as mentioned further above—we do not elaborate this case here. The metadata predicate $\Diamond(\text{“subj”}, t_s)$ specifies whether the identifier is to be established for the party itself or a third party, where t_s is the term referring to the subject in A ’s local representation.

5.2.1.2 Output

For the discussion of the output, let $\hat{\kappa}^{gc}$ be the public, or shared, part of the cryptographic material representing the established pseudonym. Technically, $\hat{\kappa}^{gc}$ is a discrete logarithm-based group commitment of a private key of A generated as part of this registration protocol instance. Let $\tilde{\kappa}^{gc}$ be the commitment opening information, which is, technically, exactly the private key.

Formulae. The output of party A is a new formula ϕ'_A derived from ϕ_A with the free variables instantiated with terms created during the protocol as follows: The term t'_p is derived from cryptographic material $\hat{\kappa}^{gc}$ resulting from the protocol execution as $\eta(\zeta_o(\hat{\kappa}^{gc}), \text{idfo})$ and replaces T_p of the input formula. The generated subject identifier is a constant term $t'_{sid,A} = t'_{sid,B}$, also derived from cryptographic

material generated during the protocol as outlined next. In Case (1) of a registration identifier object, the same terms $\mathbf{t}'_{h,A} = \mathbf{t}'_{s,A}$ for *holder* and *subject* for **A**'s output are derived from cryptographic material from the protocol execution by computation of $\eta(\zeta_s(\widehat{\kappa}^{\text{gc}}, \text{pty}))$, in Case (2) of a delegation registration identifier object, the *holder* term $\mathbf{t}'_{h,A}$ used by **A** is the same as the corresponding input term $\mathbf{t}_{h,A}$, that is, $\mathbf{t}'_{h,A} = \mathbf{t}_{h,A}$, and the *subject* term $\mathbf{t}'_{s,A}$ for **A**'s output is the same as the corresponding term $\mathbf{t}'_{s,B}$ for **B**'s output as explained below.

Because those technical details could not be provided in the summary of the cryptographic protocol generation in Chapter 6, we briefly introduce the abovementioned functions now. The function ζ_o computes a term for an object identifier based on applying a collision-resistant one-way function on its argument and the function η encodes its first argument as a value in the set corresponding to the type specified through its second argument. Analogously, the function ζ_s computes a term for referring to the holder of the cryptographic object argument, and ζ_d computes a term for the subject and is used only in case the subject is different to the holder.

The output of party **B** is a new formula ϕ'_B derived from ϕ_A with free variables instantiated with terms created during the protocol, similar to ϕ'_A . The *subjectId* receives the same term in either case. In Case (1), the equal *holder* and *subject* terms $\mathbf{t}'_{h,B} = \mathbf{t}'_{s,B}$ are computed like for **A**. The difference to **A**'s output is that in Case (2) the constant *subject* term $\mathbf{t}'_{s,B}$ for **B**'s output is created from cryptographic material from the protocol execution as $\eta(\zeta_d(\widehat{\kappa}^{\text{gc}}, \text{pty}))$ and the *holder* is computed as $\mathbf{t}'_{h,B} := \eta(\zeta_s(\widehat{\kappa}^{\text{gc}}, \text{pty}))$.

That is, most free variables are instantiated in an analogous manner, except for the stated differences, for both formulae, because of the protocol being a registration protocol and this not establishing any undesired linkabilities.

The differences in terms between ϕ'_A and ϕ'_B for referring to the same party defines a *term renaming function*, or *term replacement function*, ϱ^{idf} as given in (5.4). Only party **A** is privy of this term renaming function.

$$\varrho^{\text{idf}} := \begin{cases} \{\} & \text{for Case (1)} \\ \{(\mathbf{t}_{h,A}, \mathbf{t}'_{h,B})\} & \text{for Case (2)} \end{cases} \quad (5.4)$$

Metadata. Party **A** receives the following metadata as output in the form of metadata predicates: the *role* of being holder of the identifier relationship, the public and private parts $\widehat{\kappa}^{\text{gc}}$ and $\widetilde{\kappa}^{\text{gc}}$ of the cryptographic material of the identifier object of the identifier relationship that has been established, being the public and private part of the established cryptographic pseudonym, the renaming function ϱ^{idf} for the identifier object, and the subject identifier for **B** with whom the identifier relationship has been established. $\widetilde{\kappa}^{\text{gc}}$ comprises the private key of the party created in the registration interaction.⁴¹ As one example for the predicate notation, the term renaming function ϱ^{idf} is returned as metadata predicate $\diamond(\text{“renFct”}, \varrho^{\text{idf}})$ in **A**'s output.

⁴¹The private key can, in an implementation, be handled as separate entity that is accordingly secured, e.g., through containment in a tamper-resistant security token.

Party **B** receives the *role* of being recipient of the identifier relationship, the public part $\widehat{\kappa}^{\text{gc}}$ of the pseudonym, and the identifier under which it has been known towards **A** in the protocol.

B learns nothing more than its output data presented above, particularly not any further information about **A**, which is a main property of a protocol for establishing an identifier object. Particularly, party **B** does not learn the formulae $\phi_{\mathbf{A}}$ and $\phi'_{\mathbf{A}}$ during protocol execution.

5.2.1.3 Post Processing

Once the protocol has been successfully executed, entries in the parties' data repositories are created by the parties as follows: **A** creates a new entry in its identifier relationships comprising the formula $\phi'_{\mathbf{A}}$ as object specification formula of the new identifier object and the metadata predicates it has received as output. Through the *role* predicate, this identifier relationship is flagged as one of the party's identifier relationships it has established with another party. To reflect the data exchange with **B**, **A** creates an entry in its data track with itself being the subject, that is, the term $\mathbf{t}'_{\text{sid},\mathbf{A}}$ being used, and data recipient being $\mathbf{t}_{\text{sid},\mathbf{B}}$ and the revealed data being formula $\phi'_{\mathbf{A}}$. **B** creates a corresponding identifier relationship entry with formula $\phi'_{\mathbf{B}}$ and subject $\mathbf{t}'_{\text{sid},\mathbf{B}}$, flagged as the recipient side of the relationship through the output metadata. **B** also creates an entry in its profile data for subject term $\mathbf{t}'_{\text{sid},\mathbf{B}}$ based on formula $\phi'_{\mathbf{B}}$.

5.2.2 Protocol for Establishing an Identifier Relationship

For the case when a subject identifier is to be established not in the context of a registration interaction, an instance of the protocol for establishing a non-registration subject identifier is executed. This protocol is denoted as **EstIdtfRel** and is similar to the above protocol in terms of interface and cryptographic realization.

The protocol models **A** establishing a pseudonym with a party **B**, based on an already-established registration domain or delegation registration domain—cryptographically, the private keys thereof—for binding the new identifier object to the user in a cryptographic manner.

The formulae $\phi_{\mathbf{A}}$, $\phi'_{\mathbf{A}}$, and $\phi'_{\mathbf{B}}$ are similar to the formulae for the registration protocol, with the differences outlined next. The discussion on metadata applies analogously and is not repeated.

5.2.2.1 Input

As for the registration protocol, the non-registration protocol has two cases for the handling of terms depending on whether the pseudonym is to be established about the party **A** or a third party **D**, indicated through a metadata predicate in party **A**'s input. Case (1) of a regular identifier object with **A** being the subject is realized through a constant term referring to **A** being used as value for the attributes *holder* and *subject* in $\phi_{\mathbf{A}}$. Both are again the term that **A** uses in its local data representation for referring to itself in the registration domain in which to establish

the identifier object. The registration domain is implied by the subject term. Case (2) of a delegation identifier object with a third party \mathbf{D} being the subject is handled by using a constant term for the holder equally to Case (1) and a constant term for the value of the *subject* attribute that refers to \mathbf{D} as subject. This term is, analogous to Case (1), the term \mathbf{A} uses locally to refer to the pseudonym subject and has been introduced into \mathbf{A} 's data repository through a delegation registration identifier object.

For Case (1), the term used to refer to the holder and subject is the term created with the establishment of a registration identifier object the private key of which the to-be-established identifier object is to be bound to. For Case (2), this is true for the holder of the to-be-established object, while the subject term is the one created with the establishment of a registration identifier object for \mathbf{D} , the delegation private key of which the to-be-established identifier object is to be cryptographically associated with.

Output. The output formula $\phi'_{\mathbf{A}}$ is created with some notable differences to the registration protocol. The constant terms \mathbf{t}'_p and $\mathbf{t}'_{sid,\mathbf{A}} = \mathbf{t}'_{sid,\mathbf{B}}$ are created equally to the registration protocol. The holder and subject are handled differently, as explained next. For Case (1), both holder and subject are the same constant $\mathbf{t}_{h,\mathbf{A}} = \mathbf{t}_{s,\mathbf{A}}$ used in the input to indicate the sameness of the party with those functions. The constant is the one created as holder term in the establishment of the registration identifier object of the party in whose registration domain the current object is to be established. For Case (2), the holder is the constant referring to \mathbf{A} as in \mathbf{A} 's input and the subject the constant referring to \mathbf{D} in the input. Note that $\phi'_{\mathbf{A}}$ specifies the representation of the identifier relationship using \mathbf{A} 's local terms.

The output formula $\phi'_{\mathbf{B}}$ again requires further consideration and exposes differences to the registration case. \mathbf{t}'_p and $\mathbf{t}'_{sid,\mathbf{B}} = \mathbf{t}'_{sid,\mathbf{A}}$ are the same terms used in $\phi'_{\mathbf{A}}$ computed from the cryptographic material. For Case (1), the subject term $\mathbf{t}'_{s,\mathbf{B}}$ is computed as $\mathbf{t}'_{s,\mathbf{B}} := \eta(\zeta_s(\widehat{\kappa}^{\text{gc}}), \text{pty})$ from the cryptographic material generated in the cryptographic protocol, and so is the holder term $\mathbf{t}'_{h,\mathbf{B}} := \eta(\zeta_s(\widehat{\kappa}^{\text{gc}}), \text{pty})$ because both terms refer to the same party. For Case (2), the holder term $\mathbf{t}'_{h,\mathbf{B}}$ is computed as for Case (1), while the subject term is computed from the cryptographic material in a different way as $\mathbf{t}'_{s,\mathbf{B}} := \eta(\zeta_d(\widehat{\kappa}^{\text{gc}}), \text{pty})$. This results in the term $\mathbf{t}'_{h,\mathbf{B}}$ for referring to the *holder* and the term $\mathbf{t}'_{s,\mathbf{B}}$ referring to the *subject* of the identifier object being different terms to the ones in \mathbf{A} 's output formula, while $\mathbf{t}'_{sid,\mathbf{B}}$ is the same as $\mathbf{t}_{sid,\mathbf{A}}$.

The renaming of terms between $\phi'_{\mathbf{A}}$ and $\phi'_{\mathbf{B}}$ defines, as in the registration protocol, a renaming function ϱ^{idf} as presented in (5.5).

$$\varrho^{\text{idf}} = \begin{cases} \{(\mathbf{t}_{h,\mathbf{A}}, \mathbf{t}'_{h,\mathbf{B}})\} & \text{for Case (1)} \\ \{(\mathbf{t}_{h,\mathbf{A}}, \mathbf{t}'_{h,\mathbf{B}}), (\mathbf{t}_{s,\mathbf{A}}, \mathbf{t}'_{s,\mathbf{B}})\} & \text{for Case (2)} \end{cases} \quad (5.5)$$

The function is returned as metadata predicate $\Diamond(\text{"renFct"}, \varrho^{\text{idf}})$ in \mathbf{A} 's output, determining the term renaming in later protocols related to this object.

5.2.3 Multi-party and Public Pseudonyms

The interface for the protocols `EstldtfRel` allow for establishing a single subject identifier with one other party—the standard case of cryptographic pseudonyms based on credential protocols. When considering the valid case of establishing the same pseudonym with multiple other parties as, e.g., required for delegation (*multi-party identifier*) or the public (*public identifier*), that is, all parties in the system, either an extension of the above or new protocols are required.

For the credential-based delegation protocols, we require that the delegater establish the same cryptographic pseudonym with two parties. This requires an extension of both the protocol interface and the cryptographic protocol. At the interface level, the extension reflects the possibility of an already-established pseudonym being also established with another party. Cryptographically, this can be realized by communicating the public part $\hat{\kappa}^{\text{gc}}$ of the cryptographic pseudonym to the new recipient and proving holdership of the corresponding private part $\tilde{\kappa}^{\text{gc}}$ without establishing a new cryptographic pseudonym.

When a party **A** creates a public identifier, or, public pseudonym, that is, a public name used by all parties for referring to **A**, it practically executes a single protocol, very similar to the `EstldtfRelReg` or `EstldtfRel` protocol, in which it creates the private and public formulae and cryptographic objects for the pseudonym and makes the public part available to all parties in the system. The latter can be achieved by standard means of uploading the public part to a public pseudonym repository. With the uploading, no pseudonym has yet been established by **A** with any other party—this happens only later once another party obtains the public part of the pseudonym from the repository.

A special case of this are public keys of parties used on the Internet as of today, this special case being one of the motivations for introducing public pseudonyms as they allow us to model identifier objects with service providers and certifiers as subjects in exactly the same way as the identifier objects of users. Those of service providers and certifiers are based on conventional PKI technology, though, not credential systems, today.

The same terms for the subject and holder as defined through the renaming function for the identifier object are exposed towards all parties the identifier object is established with for a multi-party or public pseudonym.

5.2.4 Traditional Technology

For realizing PKI functionality, both registration and non-registration protocols need to be specified also for traditional certificate protocols. Using the ideas we employed for credential systems, this is an analogous task. A signature key pair of a party can be seen as a cryptographic pseudonym, much like in the case of credential systems. The terms used for referring to parties can be derived from the cryptographic material similarly to the credential protocols. See Sec. 4.8 for a discussion on expressing PKI functionality.

5.2.5 Relation to the Channel Model

Relating to the model of Chapter 3, a new identifier relationship is established whenever a party needs to interact with another party. This does not explicitly correspond to a channel transformation rule in the model. Multi-party pseudonyms are required for realizing the transformation rule for delegation, while public pseudonyms are used for realizing PKI functionality—certifiers and service providers act under public pseudonyms. Pseudonymous certifiers act under multi-party pseudonyms shared with all or a subset of all parties.

5.3 Establishing Identity Relationships

Establishing identity relationships based on credential protocols is realized through the protocol `EstIdtyRel` for a party **A** obtaining a credential with subject **A** or a delegater **D** from a certifier **B**.

5.3.1 Protocol for Establishing an Identity Relationship

The protocol `EstIdtyRel` creates a new identity relationship between a party, e.g., a user, and another party, a certifier. This comprises both registration and non-registration as well as delegation identity relationships. As explained in detail in Section 4.7.2, an identity relationship specifies an identity vouching relationship between two parties and can, once established, be used by its holder to reveal certified attribute data about the subject of the identity relationship to other parties. Establishing an identity relationship means that the certifier agrees to certify the data related to the subject in the identity relationship and enabling the holder of the identity relationship to utilize the identity relationship in future interactions with arbitrary parties for releasing (parts of) the certified attribute data contained therein. When employing private certificate protocols, an identity relationship can be used for releasing data based on the identity relationship without further involving the certifier in the data release interactions.⁴² We have found the strong integration of identity relationships and related protocols with our data model to be crucial for obtaining a practical system, thus those aspects are discussed in detail.

5.3.1.1 Languages

The protocol inputs and outputs for establishing an identity relationship based on a private certificate and the corresponding identity are described, among other parameters, through formulae expressed using our data modeling approach. This gives rise to multiple languages $\mathcal{L}^{\text{idty}}$, $\mathcal{L}^{\text{idty}/p}$, and $\mathcal{L}^{\text{idty}/r}$ required specifically in the context of identity relationships and their establishment. The language $\mathcal{L}^{\text{idty}}$, with $\mathcal{L}^{\text{idty}} \subset \mathcal{L}^g$, can be seen as the common ground for both $\mathcal{L}^{\text{idty}/p}$ and $\mathcal{L}^{\text{idty}/r}$ and is itself

⁴²Certificate revocation might in this case involve the certifier or a third party, though in a different way than an online certifier vouching for identity statements in a protocol based on online certifiers.

not applied for data representation. The language $\mathcal{L}^{\text{idy}/p}$ is used for representing the formula-based input to and output of the protocol **EstIdtyRel** for establishing an identity relationship, while $\mathcal{L}^{\text{idy}/r}$ is used for representing the identity of the resulting identity relationship. $\mathcal{L}^{\text{idy}/r}$ has been introduced already with the concept of identity relationships in Sec. 4.7.2.

$\mathcal{L}^{\text{idy}} \subset \mathcal{L}^{\text{e}}$ can be informally specified as comprising **Eq** predicates⁴³ with appropriate type signature with one argument being an attribute of the to-be-established identity, the identity being referred to through a variable or constant term, and the other argument representing an attribute value, either through a constant term, or an attribute reference of a conditionally released identity or opaque identity referred to through a constant. The predicates are connected only through the \wedge -connective throughout the formula. Predicates on the *holder*, *subject*, *certifier*, *type*, and *protocol* attributes of the identity must be stated, and the *holder*, *subject*, and *certifier* attributes must be related through the logical equality operator $=$ to a constant term. Further predicates specifying the certifier through a certifier identity can be present.

5.3.1.2 Prerequisites

Multiple aspects need to be discussed as prerequisites to the interface for the protocol **EstIdtyRel**.

Identifier objects. As prerequisites for executing the protocol for establishing an identity relationship, **A** needs to have an identifier object \mathbf{t}_p established with the certifier for specifying the holder of the new identity, as well as an identifier object $\mathbf{t}_{p'}$ for the subject, where the latter can be the same one as the further.

Prerequisite protocol steps. For the case of establishing a *non-delegation identity*, it is assumed that **A** authenticates a formula ϕ to **B** using **RelData**.

For the case of establishing a *delegation identity*, protocol steps for initiating the delegation as discussed in Chapter 3 are assumed to be executed as follows. **D** initiates the delegation with the certifier **B** and establishes the identifier object $\mathbf{t}_{p'_{B|A}}$ with it. The corresponding pseudonym in the channel model is $\mathcal{D}_{c_{k+1}|a}$. We also assume that **A** establishes the identifier object \mathbf{t}_p with **D**. The delegator establishes and authenticates the pseudonym $\mathbf{t}_{p'_{B|A}}$ also towards **A** and signs the delegation authorization token \mathcal{Y} it sends to **A** with the corresponding authenticated statement. The token contains the pseudonym $\mathbf{t}_{p'_{B|A}}$ of **A** with **D**, that is, $\mathcal{A}_{c_{k+1}|d}$ in the channel model.

A provides the delegation authorization token \mathcal{Y} to **B** to prove eligibility for being the holder of the to-be-established delegation identity relationship. **A** authenticates towards **B** under \mathbf{t}_p through the prove of ϕ . Over the authenticated channel **A**

⁴³For the extension discussed in Sec. 4.4.17, also the inequality predicates **Geq**, **Gt**, **Leq**, and **Lt** can be applicable for specifying the values of user attributes of the identity. The **Neq** predicate is not particularly useful in this context and therefore not further discussed, though, could be employed for specifying values of user attributes.

sends \mathcal{Y} and \mathbf{B} observes the same pseudonym \mathbf{t}_p contained therein as the one to the holder of which \mathbf{B} has been instructed by \mathbf{D} to issue the delegation certificate. Through this it is ensured that the certificate is issued to right party \mathbf{A} and is, cryptographically, bound to the right private key of \mathbf{A} on which also \mathbf{t}_p is based.

Building on those assumptions on required protocol steps for a non-delegation or delegated identity relationship having been executed, the further discussions elaborate on the establishment of the identity (relationship) between \mathbf{A} and \mathbf{B} with the subject being \mathbf{A} or the delegater \mathbf{D} .

Prerequisite formula. The abovementioned formula ϕ making pseudonym assertions has been proven by \mathbf{A} to \mathbf{B} . We give details on the formula and particularly the pseudonym assertions made, using the terms in the view of \mathbf{B} . For the non-delegation case, holdership of the identifier object \mathbf{t}_p and $\mathbf{t}_{p'}$ is proven, both of which must have the same term $\mathbf{t}_{h,p,\mathbf{A}}$ as their holder and subject and $\mathbf{t}_{sid,p}$ as a subject identifier (see (5.6)).⁴⁴ In this case, the identifier objects are either registration or derived identifier objects. For the delegation case, holdership of two distinct identifier objects \mathbf{t}_p and $\mathbf{t}_{p'}$ needs to be proven. The holder and subject of \mathbf{t}_p are $\mathbf{t}_{h,p,\mathbf{A}}$ and $\mathbf{t}_{h,p,\mathbf{A}}$, respectively, and of $\mathbf{t}_{p'}$ the terms $\mathbf{t}_{h,p',\mathbf{A}}$ and $\mathbf{t}_{s,p'}$. Both $\mathbf{t}_{h,p,\mathbf{A}}$ and $\mathbf{t}_{h,p',\mathbf{A}}$ refer to the same party indicated through equality in the logic, while $\mathbf{t}_{s,p'}$ refers to a delegater \mathbf{D} (see (5.7)). In this case, \mathbf{t}_p is a registration or derived identifier object with \mathbf{A} as subject, and $\mathbf{t}_{p'}$ a delegation registration identifier object with delegater \mathbf{D} as subject. The terms $\mathbf{t}_{sid,p}$ and $\mathbf{t}_{sid,p'}$ are the pseudonym identifiers of the identifier objects and are used in the protocol interface of the protocol `EstIdtyRel` for unambiguously specifying the holder and subject of the to-be-established identity, also in case of assertions over more than two identifier objects being made in ϕ .

$$\begin{aligned}\phi_n &= \mathbf{t}_p.\text{holder} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \mathbf{t}_p.\text{subject} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \\ &\quad \text{Eq}(\mathbf{t}_p.\text{subjectId}, \mathbf{t}_{sid,p}) \wedge \mathbf{t}_{p'}.\text{holder} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \\ &\quad \mathbf{t}_{p'}.\text{subject} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \text{Eq}(\mathbf{t}_{p'}.\text{subjectId}, \mathbf{t}_{sid,p}) \wedge \tilde{\phi}_n\end{aligned}\tag{5.6}$$

$$\begin{aligned}\phi_d &= \mathbf{t}_p.\text{holder} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \mathbf{t}_p.\text{subject} = \mathbf{t}_{h,p,\mathbf{A}} \wedge \\ &\quad \text{Eq}(\mathbf{t}_p.\text{subjectId}, \mathbf{t}_{sid,p}) \wedge \mathbf{t}_{p'}.\text{holder} = \mathbf{t}_{h,p',\mathbf{A}} \wedge \\ &\quad \mathbf{t}_{p'}.\text{subject} = \mathbf{t}_{s,p'} \wedge \text{Eq}(\mathbf{t}_{p'}.\text{subjectId}, \mathbf{t}_{sid,p'}) \wedge \\ &\quad \mathbf{t}_{h,p,\mathbf{A}} = \mathbf{t}_{h,p',\mathbf{A}} \wedge \tilde{\phi}_d\end{aligned}\tag{5.7}$$

The assertions over identifier objects are part of the identifier object introduction sub-formula of an identity statement ϕ_n or ϕ_d of language $\mathcal{L}^{p/c}$ of Sec. 4.5. Let such statement be denoted by ϕ , with $\phi \in \mathcal{L}^{p/c}$. Let the sub-formula $\tilde{\phi}$ be the remaining part of such comprehensive identity statement comprising the sub-formulae beyond the identifier object statements. A formula ϕ can be any formula allowed in the protocol for releasing data that is provable with our cryptographic protocols and

⁴⁴Technically, this can be realized by a single identifier object being proven, the complications here are of purely syntactic nature and arise from two identifier objects being exposed at the protocol interface for it capturing also the delegation case.

can particularly also contain \vee -connectives for the attribute assertions as far as this is useful from the perspective of the use case.

Additional identifier objects may be referred to in ϕ , e.g., when objects from multiple bridged registration domains are referred to, though, are not further relevant in the context of establishing an identity relationship. Terms \mathbf{t}_p and $\mathbf{t}_{p'}$ refer to two distinct identifier objects chosen from all identifier objects in ϕ to represent the holder and subject of the new identity.⁴⁵

Multiple identity statements can be made through means of multiple identity statement formulae as long as the above-presented pseudonym assertions are available and the different statements bound accordingly to the same party through identifier objects. We assume in the remainder, without loss of generality, that a single formula ϕ has been revealed and proven by **A** to **B**. This can be easily generalized to multiple formulae proven in the current session or different sessions, with statements about **A**, **D**, and possibly other parties, and stored in the profile data of **B**.

As explained in Sec. 5.4, for the protocol **RelData** for proving—or releasing—data statements, a formula $\phi \in \mathcal{L}^{p/c}$ related to such protocol instance refers to objects holdership of which is proven and which may be created during the data release protocol. Let the objects holdership of which is proven, regardless of whether they have been generated in the data release protocol, be the following: the primary identifier objects $\mathbf{t}_1^{\text{idf}}, \dots, \mathbf{t}_{\ell^{\text{idf}}}^{\text{idf}}$, primary identities $\mathbf{t}_1^{\text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\text{id}}$, opaque identities $\mathbf{t}_1^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$, and conditionally released identities $\mathbf{t}_1^{\text{crid}}, \dots, \mathbf{t}_{\ell^{\text{crid}}}^{\text{crid}}$. Note that none of the identities is referred to in the current instance of the protocol for establishing an identity relationship, only the other mentioned objects are.

It is important to observe that **A** uses different terms than those exposed to **B** as part of its view on formula ϕ for referring to holders and subjects as well as the primary identities in the formula—see Sec. 4.8 and Sec. 5.4 for details on the renaming of terms in the context of the **RelData** protocol. The same term renaming as for ϕ applies to the terms reoccurring in the interface of the protocol for establishing an identity relationship, and applies to both the formulae and terms outside of formulae referring to objects. Let the local term used by **A** for $\mathbf{t}_{h,p,\mathbf{A}}$ be $\bar{\mathbf{t}}_{h,p,\mathbf{A}}$ and for $\mathbf{t}_{s,p'}$ be $\bar{\mathbf{t}}_{s,p'}$.

Constructing the issuing specification. For obtaining the formula $\phi_{\mathbf{B}} \in \mathcal{L}^{\text{id}/p}$ specifying the certificate issuing protocol, **B** constructs a new formula $\hat{\phi}^{\text{prf}}$ by combining the prerequisite formula ϕ proven by **A** and an additional specification formula ϕ^{prf} .

Thereby, the certifier **B** makes, through $\hat{\phi}^{\text{prf}}$, further attribute assertions, e.g., related to access rights to a service, about the subject **A** or **D** of the to-be-generated identity. Technically, $\hat{\phi}^{\text{prf}}$ is expressed in a simplified fragment of the language $\mathcal{L}^{p/c}$ sketched next, which is similar to the language $\mathcal{L}^{\text{id}/r}$ for expressing identity

⁴⁵In a practical system, establishing the identifier objects \mathbf{t}_p and $\mathbf{t}_{p'}$ or further identifier objects can be integrated with the prerequisite statement ϕ to be proven by **A** to **B**. For obtaining a cleaner formalism, we assume in this thesis that those protocols are standalone protocols.

specification formulae of identity relationships. A formula in this language expresses the certifier through a third-party specification sub-formula and attribute predicates over one or more identities or identifiers or combinations thereof. The attribute predicates in the formula comprise as a main part the statements that **B** makes about the subject of the to-be-issued identity in addition. The identity is referred to through a constant term derived in a well-defined manner from the predicates specified over it. The certifier of the identity is specified to be **B**, using a term **B** uses for referring to itself in its local data representation. This formalizes that the statements are declarations by party **B**. The holder and subject of the identity are specified to be the terms used in t_p and $t_{p'}$ in ϕ .

In the case of establishing a delegation identity, the formula $\tilde{\phi}^{\text{prf}}$ comprises the attributes of the identity to be delegated and the new identity is associated with the subject being the delegater.

An extension to delegating also attributes that have originally been issued based on opaque or conditionally released identities can be achieved by the delegater providing the private cryptographic material and object specification formulae of those objects to the delegatee and the delegatee using them as input to the protocol. This needs to be considered in the prerequisite steps for establishing a delegation. We do not provide the details for this, though, the cryptographic protocol for issuing is technically analogous to that for issuing certificates on attributes of opaque or conditionally released identities the party has established itself with the certifier with the additional prerequisite steps having been carried out.

The formula ϕ and $\tilde{\phi}^{\text{prf}}$ are combined through conjunction $\hat{\phi}^{\text{prf}} = \phi \wedge \tilde{\phi}^{\text{prf}}$. Thereby, $\tilde{\phi}^{\text{prf}}$ adds additional attribute statements to the attribute assertions sub-formula of ϕ . When conjuncting the third-party specification formula of $\tilde{\phi}^{\text{prf}}$ with that of ϕ and doing the same for the attribute assertion sub-formulae, the resulting formula $\hat{\phi}^{\text{prf}}$ meets the syntax requirements of $\mathcal{L}^{p/c}$ and thus $\hat{\phi}^{\text{prf}} \in \mathcal{L}^{p/c}$.

The formula $\hat{\phi}^{\text{prf}}$ expresses potentially excessive information in addition to what is to be represented through the new identity. A transformation \mathcal{F} is applied by the certifier **B** to $\hat{\phi}^{\text{prf}}$ for obtaining the formula $\phi_{\mathbf{B}}$. This transformation is parameterized and depends on the policy of the certifier specifying which information that has flown from ϕ to $\hat{\phi}^{\text{prf}}$ to retain in $\phi_{\mathbf{B}}$. Based on the parameters, the transformation realizes the following: An equality predicate in the attribute assertion sub-formula of $\hat{\phi}^{\text{prf}}$ may yield an equality predicate over the same or a parameter-determined attribute of the to-be-established identity T_c and the same constant value or attribute of a conditionally released or opaque identity. That is, predicates expressed over different identities in $\hat{\phi}^{\text{prf}}$ that may be referred to in the predicates of $\hat{\phi}^{\text{prf}}$ are now expressed over a single variable T_c for the new identity to be established. The holder and subject of T_c are specified to be equal to the holder of t_p and subject of $t_{p'}$, respectively. The certifier of T_c is specified to be equal to **B** through the term used also in a certifier identity with subject **B** that can be obtained by parties who will later verify proofs over the new identity to be created. Further certification metadata is added to the identity using equality predicates. An equality predicate on the attribute *type* of T_c is added with the value being the constant representing the type of the new identity.

Only \wedge -connected predicates are allowed in the formula as it specifies a new identity relationship.⁴⁶ Only a subset of the predicates of $\hat{\phi}^{\text{prf}}$ is retained in $\phi_{\mathbf{B}}$, the remaining predicates may have been required for fulfilling the authorization policy of \mathbf{B} for allowing \mathbf{A} to establish a new identity relationship with \mathbf{B} .

The function \mathcal{F} is tightly dependent on the intentions of \mathbf{B} for the identity relationship to be established and cannot be fully formalized in general. For example, \mathbf{B} may accept an attribute with a certain name in ϕ , though, re-certify it in \mathbf{T}_c under another attribute name, which can be the case when different certifiers use different attributes to refer to the same conceptual entity, e.g., a last name.

A technical realization of the function \mathcal{F} in a practical system can be obtained through the concept of a *template* that specifies the mapping from a formula ϕ and additional attributes representing $\hat{\phi}^{\text{prf}}$ obtained from data sources at the party to a formula $\phi_{\mathbf{B}}$ as discussed in [Som11].

Inequality predicates as explained in the extension to the data model in Sec. 4.4.17 can be used in ϕ , $\hat{\phi}^{\text{prf}}$, and $\hat{\phi}^{\text{prf}}$ as an extension to the process for establishing identity relationships for identities with attributes having an inequality as operator.

Inverse renaming. As certain terms have been renamed by \mathbf{A} through the RelData protocol and those renamed terms are used by \mathbf{B} for representing the issuing specification formula $\phi_{\mathbf{B}}$ in the EstIdtyRel protocol, \mathbf{A} needs to perform a reverse renaming of terms in $\phi_{\mathbf{B}}$ in order to obtain an intermediate formula $\phi''_{\mathbf{A}}$ based on its locally used terms. We denote the renaming transformation as κ^{-1} . Recall that the holder and subject are defined through the identifier objects \mathbf{t}_p and $\mathbf{t}_{p'}$ of ϕ , respectively.

\mathbf{A} receives from \mathbf{B} the specification formula $\phi_{\mathbf{B}}$ and the pseudonym identifiers $\mathbf{t}_{\text{sid},h}$ denoting the holder of the to-be-established identity and $\mathbf{t}_{\text{sid},s}$ denoting its subject as part of the protocol. The party retrieves the renaming functions $\varrho_{\mathbf{t}_p}^{\text{idf}}$ and $\varrho_{\mathbf{t}_{p'}}^{\text{idf}}$ from the metadata of the corresponding identifier relationships of \mathcal{R}^{idf} using the *subjectId* values $\mathbf{t}_{\text{sid},h}$ and $\mathbf{t}_{\text{sid},s}$ for the lookup. It applies the inverse functions $(\varrho_{\mathbf{t}_p}^{\text{idf}})^{-1}$ and $(\varrho_{\mathbf{t}_{p'}}^{\text{idf}})^{-1}$ to the holder and subject terms in the received formula $\phi_{\mathbf{B}}$ that are elements of the preimages of the inverse renaming functions. The inverse of the discrete function is obtained in a straightforward way by replacing every tuple x in the function's set-based specification through $(\pi^2(x), \pi^1(x))$. The renaming leads to the mapping of terms between $\phi_{\mathbf{B}}$ and $\phi''_{\mathbf{A}}$ as defined through the mentioned renaming functions.

This renaming establishes the terms the party uses locally to refer to itself and, if applicable, the delegater in the formula. The protocol specification formula $\phi''_{\mathbf{A}}$ obtained through applying κ^{-1} on $\phi_{\mathbf{B}}$ is the input of \mathbf{A} for the cryptographic protocol for issuing a credential, while \mathbf{B} 's input for the cryptographic protocol is $\phi_{\mathbf{B}}$.

The inverse renaming κ^{-1} essentially reverses the renaming of terms performed

⁴⁶For the credential protocols based on any of the Camenisch–Lysyanskaya signature protocols, only the conjunction operator is allowed for specifying the attribute statements of a to-be-issued credential.

in the instance of the **RelData** protocol for releasing ϕ . The required mappings are comprised in the renaming functions $\varrho_{t_p}^{\text{idf}}$ and $\varrho_{t_{p'}}^{\text{idf}}$ for the primary identifier objects specifying the holder and subject for T_c referred to in ϕ . We refer the reader to the general specification of the renaming in Sec. 4.8 and the discussion on the data release protocol in Sec. 5.4 later in this chapter.

Object equality predicates between terms as expressed through the $=$ -relation of the language can be left in the formula as an equality statement may now—through the term renaming—relate the same terms with object equality and thereby be a tautology. Opaque and conditionally released identities referred to in ϕ_B do not require a renaming as they are shared only between **A** and **B**. Opaque identities allow for establishing an identity relationship comprising attributes the certifier does not learn.

The formula obtained through this renaming is the one based on which party **A** executes its part of the cryptographic protocol. It is, particularly, also used for retrieving objects required as protocol inputs referred to in the formula from its local data repository.

5.3.2 Protocol Interface

The following is the protocol interface for the protocol **EstIdtyRel**, executed between a party **A** and a party **B**, the latter being the certifier, where the main input ϕ_B has been discussed in great detail further above.

EstIdtyRel protocol interface:

Input	
A	—
B	$\phi_B, t_{sid,h}, t_{sid,s}, [t_{sid,p'_B A}], \{(\mathbf{o}_i, \{\nu_{l_{B,i}}\}_{1 \leq l_{B,i} \leq n_{B,i}})\}_{1 \leq i \leq r_B}$
Output	
A	$\phi'_A, \{(\mathbf{o}'_j, \{\nu'_{l'_{A,j}}\}_{1 \leq l'_{A,j} \leq n'_{A,j}})\}_{1 \leq j \leq r'_A}, success_A$
B	$\phi'_B, \{(\mathbf{o}'_k, \{\nu'_{l'_{B,k}}\}_{1 \leq l'_{B,k} \leq n'_{B,k}})\}_{1 \leq k \leq r'_B}, success_B$

The *input* is explained first. The protocol input is determined by the certifier **B** because it is the party that decides on the data to be included in the identity relationship that it will vouch for. Clearly, this can be a function on the statements received in this or previous interactions with **A** or data obtained through other means by **B**. **A** does not provide input to the protocol at the level of the protocol interface.

The input of **B** comprises a data statement ϕ_B , constant terms $t_{sid,h}$ and $t_{sid,s}$, and metadata predicates expressed on the formula and objects referred to from the formula. The term $t_{sid,h} :: \text{idf}$ is the identifier the prospective holder **A** of the identity relationship is known to **B** under. This identifier must always be known by **B** for the following reasons: An identity relationship is always established based on one or two identifier objects its recipient holds. Also, **B** needs to know **A** by some identifier, e.g., for purposes of allowing for revocation of **A**'s anonymity as well as for non-credential-based protocols where **B** needs to be involved online when **A** makes

use of the identity relationship to release data. The optional argument $\mathbf{t}_{sid,p'_B|A}$ defines the subject identifier that the delegater has used for initiating the delegation and that is comprised in the delegation authorization token \mathcal{Y} . The metadata being required depend on the protocol used for establishing the identity relationship. Most metadata, such as public keys, required by the protocol need, in an implementation of the system, not be passed at the API level, but can be obtained from the protocol implementation from other components of the architecture or remote services, which leads to a simplification of the interface.

The *output* of \mathbf{A} comprises a formula ϕ'_A as well as a set of metadata predicates returned at the side of \mathbf{A} . \mathbf{B} 's output comprises, analogously, a formula ϕ'_B and metadata predicates.

We next give details on the parameters of the interface and particularly show the differences between the output and the input as there are some important aspects to note when using private certificate systems as underlying protocols. The elements that hide most of the interface complexity are the formulae at both of the parties.

5.3.2.1 Input formula

A formula ϕ_B specifying the identity relationship to be created is expressed in $\mathcal{L}^{\text{idy}/p}$. The language is based on \mathcal{L}^{idy} with the following constraints: the identity is always referred to through a variable for expressing that the underlying object be created in the protocol. Conceptually, the input formula stands—as it contains free variables—for the set of valid output formulae of the protocol, while the output formulae are statement-type formulae without free variables.

$$\begin{aligned}
 \phi_B = & \mathbf{T}_c.\text{certifier} = \mathbf{t}_{s_B} \wedge \mathbf{t}_{cid}.\text{subject} = \mathbf{t}_{s_B} \wedge \tilde{\phi}_{cid} \wedge \\
 & \mathbf{T}_c.\text{holder} = \mathbf{t}_{h,p,A} \wedge \mathbf{T}_c.\text{subject} = \mathbf{t}_{s,p'} \wedge \\
 & \text{Eq}(\mathbf{T}_c.a_1, \mathbf{t}_1^a) \wedge \dots \wedge \text{Eq}(\mathbf{T}_c.a_u, \mathbf{t}_u^a) \wedge \\
 & \text{Eq}(\mathbf{T}_c.a_1^\diamond, \mathbf{t}_1^{a_1^\diamond}) \wedge \dots \wedge \text{Eq}(\mathbf{T}_c.a_q^\diamond, \mathbf{t}_q^{a_q^\diamond})
 \end{aligned} \tag{5.8}$$

The formula structure for ϕ_B as presented in (5.8) is explained next. The certifier specification (line 1 of (5.8)) comprises the association of the new identity \mathbf{T}_c with an existing certifier identity, or identifier relationship, or combination of identifiers and identifier relationships, as discussed for the language $\mathcal{L}^{p/c}$ in Sec. 4.5, through a constant term \mathbf{t}_{s_B} and a subformula $\tilde{\phi}_{cid}$ detailing the certifier specification. Based on whether a non-delegation or delegation identity relationship is to be established, the value of the holder and subject of the to-be-established identity are specified as follows (line 2 of (5.8)): in the non-delegation case, the holder and subject are specified to equal $\mathbf{t}_{h,p,A}$, in the delegation case, the holder equals $\mathbf{t}_{h,p,A}$, while the subject equals $\mathbf{t}_{s,p'}$. This properly links the holder and subject to the holders and subjects of the pseudonyms holdership of which has been proven beforehand through ϕ .

The core of the formula modeling the attribute data for the to-be-generated identity comprises a sequence of \wedge -connected predicates $\text{Eq}(\mathbf{T}_c.a_i, \mathbf{t}_i^a)$ with $1 \leq i \leq u$,

each stating an equality assertion between a user attribute reference of the identity ($T_c.a_i$) and the value t_i^a it is to hold which can be either of the following: a constant term or an attribute reference of an opaque identity or of a conditionally released identity (line 3 of (5.8)). The opaque and conditionally released identities permitted to be referred to are, w.l.o.g. with simplified indexing, $t_1^{\text{oid}}, \dots, t_{\ell^{\text{oid}}}^{\text{oid}}$ and $t_1^{\text{crd}}, \dots, t_{\ell^{\text{crd}}}^{\text{crd}}$, respectively, of ϕ , which are a subset of those referred to in ϕ . Another sequence of \wedge -connected predicates specifies the in-formula metadata, comprising the temporal validity of the identity, protocol suite to be used, and possibly further attributes, expressed through predicates $\text{Eq}(T_c.a_i^\diamond, t_i^{a_i^\diamond})$ for $1 \leq i \leq q$ (line 3 of (5.8)).

The input formula ϕ_B needs to be established before protocol execution can begin. In our architecture, the certifier can use a *template* that acts as a blueprint for each new identity relationship for a class of identity relationships, e.g., for each relationship for an identity of a certain identity type.

5.3.2.2 Output formulae

The formula ϕ'_A is expressed in $\mathcal{L}^{\text{id}/r}$ and represents **A**'s view on the established identity relationship. The language is based on \mathcal{L}^{id} and similar to $\mathcal{L}^{\text{id}/p}$, though, only constant terms are allowed. All values of attributes are constants representing those values, without the option of attribute references of cryptographic objects.⁴⁷ The certification metadata does not comprise the predicates related to the certifier specification—only the predicate associating a term for the certifier with the new identity.

For obtaining the output formula ϕ'_A of **A**, the intermediate formula ϕ''_A as used by **A** as the specification for the cryptographic protocol needs to undergo multiple changes. Once the cryptographic protocol has been successfully executed, **A** computes the constant term t'_c as $\eta(\zeta_o(\tilde{\kappa}^{\text{crt}}), \text{id})$ based on the credential $\tilde{\kappa}^{\text{crt}}$ it has obtained in the protocol. Note that this computation can be based on the almost-final credential, which both issuer and recipient have access to to allow both to compute it—we abstract from this in the formal notation, though. This term is henceforth used by **A** to refer to the identity locally, e.g., in input formulae to the protocol **RelData** for releasing data and particularly to refer to the identity in ϕ'_A in place of T_c in ϕ''_A . The term is computed analogously by **B** and used by it for referring to the identity in its local identity relationship. Note that **B**'s representation of the identity relationship is not required for **A** for using the identity relationship, though, it is needed for requesting revocation of the identity t'_c .

The terms referring to the holder and subject of t'_c have been renamed already when obtaining ϕ''_A from ϕ_B to reflect **A**'s terms. This is, for example, crucial in retaining **A**'s capabilities of using deduction in our logic calculus for computing a formula for fulfilling a data request.

Another crucial change for transforming ϕ''_A to ϕ'_A is the replacement based on function ϖ of attributes specified through reference to attributes of opaque or

⁴⁷The only exception to only terms being allowed are values of user attributes which may be variables in the case of restriction formulae. This is not applicable to a formula ϕ'_A , though, only to formulae defined in addition to ϕ'_A for the identity relationship.

conditionally released identities with the corresponding attribute values, that is, constant terms, only available to **A**. The subformula specifying the attributes of t'_c is transformed to properly realize attribute references to attributes of opaque or conditionally released identities as constant terms. Each predicate $\text{Eq}(t'_c.a_i, t_o.a_{i'})$ where t_o refers to an opaque or conditionally released identity the attribute $a_{i'}$ of which the attribute a_i of the new identity is to be equal to is replaced with a new predicate. The latter states an equality predicate between $t'_c.a_i$ and the attribute value of $t_o.a_{i'}$ known to **A** as part of the private cryptographic item, $\tilde{\kappa}^{\text{dfc}}$ or $\tilde{\kappa}^{\text{enc}}$ for t_o being a cryptographic commitment or ciphertext of a verifiable encryption protocol, respectively, that is, corresponding to an opaque or a conditionally released identity, corresponding to t_o , but not known to **B**. Only those plaintext attributes of the opaque or conditionally released identities are relevant when using the established identity relationship later. Information on how the attributes were obtained, e.g., through which specific opaque identity, is not relevant for the holder of the relationship in order to release data based on it. The attribute values are required to be known in the identity relationship of **A** in order to match it against data requests for its use. Note that, optionally, **A** can obtain the formula $\phi''_{\mathbf{A}}$ as metadata output for tracking the details on how the identity relationship has been established. Predicates related to attributes of opaque or conditionally released identities can of course not be changed in $\phi'_{\mathbf{B}}$ as the corresponding attributes are hidden towards party **B**.

The predicates related to the specification of the certifier identity are used for obtaining a specification formula specifying the certifier identity. The certifier identity is returned as a metadata predicate based on which a new profile data entry of **A** is generated in the post-processing stage of the protocol. The certifier predicates are removed from the output formula as they are held separately by both parties. The certifier identity corresponds to a public key certificate for the certifier in case PKI mechanisms are used for ensuring authenticity of the certifier.

The above-discussed changes to $\phi''_{\mathbf{A}}$ for obtaining the formula $\phi'_{\mathbf{A}}$ are expressed through the function $\mathcal{F}^{\mathbf{R}}$ party **A** executes in the relevant channel transformation rules in the model of Chapter 3.

The formula $\phi'_{\mathbf{B}}$ is expressed in a fragment of $\mathcal{L}^{\text{id}/\text{p}}$ with all terms being constants. The most noteworthy difference to $\phi'_{\mathbf{A}}$ is that it can comprise attribute references to attributes of conditionally released and opaque identities instead of only constants as in $\phi'_{\mathbf{A}}$ for representing the values of user attributes. The discussed changes in the formulae are protocol-dependent and applicable for credential protocols.

5.3.2.3 Metadata

The following *on-formula metadata* can be expressed on any of the formulae $\phi_{_}$ through metadata predicates, that is, outside of our data model. Thus, such on-formula metadata do not have meaning in the data model and authorization policies cannot build on such data, and they are not considered in syntactical derivations in our logic. Such metadata is input by **B** and output to both **A** and **B** and are

discussed next.

A *technical specification* of the identity relationship specifies aspects that are required by the underlying cryptographic protocols in addition to what is formally expressed within the formula ϕ_- . For our credential protocols, an important part of this technical specification is the *certificate structure*, first discussed by the author in earlier work [CSZ06b, CSZ06a] and in Sec. 5.1.1. The certificate structure is input as metadata on the variable term referring to the to-be-created identity of $\phi_{\mathbf{B}}$ and output as metadata on the constant $\mathbf{t}'_{\mathbf{c}}$ referring to the created identity in both $\phi'_{\mathbf{A}}$ and $\phi'_{\mathbf{B}}$.

Another crucial kind of metadata input are the cryptographic objects underlying the conditionally released and opaque identities $\mathbf{t}_1^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ and $\mathbf{t}_1^{\text{crd}}, \dots, \mathbf{t}_{\ell^{\text{crd}}}^{\text{crd}}$ referred-to in $\phi_{\mathbf{B}}$. For each opaque identity $\mathbf{t}_i^{\text{oid}}$, a metadata predicate

$$\diamond(\mathbf{t}_i^{\text{oid}}, \text{"pubCryptoMaterial"}, \widehat{\kappa}^{\text{dfc}})$$

and for each conditionally released identity $\mathbf{t}_j^{\text{crd}}$, a metadata predicate

$$\diamond(\mathbf{t}_j^{\text{crd}}, \text{"pubCryptoMaterial"}, \widehat{\kappa}^{\text{enc}})$$

is required as input to the cryptographic protocol for both parties \mathbf{A} and \mathbf{B} . For \mathbf{A} , the opaque identities and conditionally released identities

$$\begin{aligned} &\diamond(\mathbf{t}_i^{\text{oid}}, \text{"privCryptoMaterial"}, \widehat{\kappa}^{\text{dfc}}) \text{ for } 1 \leq i \leq \ell^{\text{oid}} \text{ and} \\ &\diamond(\mathbf{t}_j^{\text{crd}}, \text{"privCryptoMaterial"}, \widehat{\kappa}^{\text{enc}}) \text{ for } 1 \leq j \leq \ell^{\text{crd}} \end{aligned}$$

are required as additional private cryptographic input objects. Note that those public cryptographic materials can be part of the metadata of ϕ or of formulae proven previously by \mathbf{A} to \mathbf{B} . Using their term names $\mathbf{t}_i^{\text{oid}}$ and $\mathbf{t}_j^{\text{crd}}$, they can be easily retrieved by both parties from the metadata of those previously executed protocols. We note that those input elements are obtained by the component executing the protocol instead of being explicitly passed for the reason that \mathbf{A} is not able to pass them as input as it does not know the issuing specification at the time it needs to invoke the protocol and pass the input elements. The private certificate created through the protocol is a metadata predicate output on the side of \mathbf{A} as private cryptographic material. Further on-formula metadata can be expressed as required for an implementation of a system, without giving all the details here.

Further inputs are the cryptographic materials related to the identifier objects \mathbf{t}_p and $\mathbf{t}_{p'}$ determining the registration domain and delegation registration domain of the new identity and thus the underlying private keys: \mathbf{A} inputs both private and public parts thereof, while \mathbf{B} only inputs the public parts. Those cryptographic elements are required for establishing the cryptographic binding of the issued credential to the private keys and registration domains related to \mathbf{t}_p and $\mathbf{t}_{p'}$.

The specification formula related to the certifier identity and the public key related to it are comprised in further metadata predicates. This is used for creating profile data entries on the side of \mathbf{A} for specifying the certifier of the new identity through an identity specification formula for the certifier identity corresponding

to its public key certificate and an identifier object corresponding to the public key, both using the same term to refer to the certifier. See Sec. 4.8.1 for details on how a certifier is represented by other parties through identities or identifier relationships. This is required for making automated derivations referring to the established identity, e.g., for matching the party's portfolio with data requests in the context of instances of the RelData protocol.

5.3.3 Post Processing

Once the protocol for establishing an identity relationship has been successfully executed, **A** stores ϕ'_A and the metadata predicates returned by the protocol in a newly created identity relationship flagged with a metadata predicate as an identity relationship it is holder of. Possible further object specification formulae specifying less information on the new identity can be added by **A**, e.g., through using a policy editor, for associating different authorization policies governing the release of certain attribute information in a negotiation protocol. **B** stores its formula ϕ'_B and the obtained metadata in an identity relationship flagged as one it vouches for. It needs to use the identity relationship when being involved in providing certified identity data based on the identity relationship to a data recipient in protocols with on-line issuer and for a potential revocation of the relationship in the case of credential protocols. For both parties, the new identity relationship is associated with the identifier relationships it has been issued on through metadata predicates. The input data to the protocol are not required any more after the protocol execution, yet can be retained for reasons of accountability. To summarize, the obtained identity relationships have a completely different meaning to the certifier and to the certifiee.

A creates a new profile data entry flagged as being that related to a certifier of one of its identity relationships. The formulae of the entry are the ones specifying the identity and identifier object of the certifier. The associated cryptographic metadata are a public key certificate and corresponding public key required for executing RelData protocol instances for proving holdership of the credential underlying t'_c .

5.3.4 Relation to the Channel Model

We first discuss the case of establishing a non-delegation identity. The formula ϕ_B corresponds to $\mathcal{F}(\hat{\phi}^{\text{prf}}, \mathbf{C}_i, \aleph_i)$, where $\hat{\phi}^{\text{prf}} = \phi \wedge \phi^{\text{prf}}$ in the model, which forms the basis for establishing one identity relationship with certifier \mathbf{C}_i as formalized in Rule Auth-Cred of Chapter 3 for k identity relationships. The formula ϕ_i in the model corresponds to the formula $\hat{\phi}^{\text{prf}}$ with amendments by the certifier already contained. Thus, $\hat{\phi}^{\text{prf}}$ corresponds to a combination of possibly multiple authenticated statements on the channel from **A** to **B**. The formula $\tilde{\phi}^{\text{prf}}$ thereby is authenticated in a trivial way in that it is a claim by **B** towards itself. The function \mathcal{F} models the transformation of $\hat{\phi}^{\text{prf}}$ to ϕ_B through **B**. The context \aleph_i of the certifier thereby captures the parameterization of the function \mathcal{F} to determine how the new formula is computed based on the statement authenticated to the certifier.

To obtain $\phi''_{\mathbf{A}}$ first from $\phi_{\mathbf{B}}$ and $\phi'_{\mathbf{A}}$ in a second step corresponds to \mathbf{A} applying the function $\mathcal{F}^{\mathbf{R}}$ in the channel model to express \mathbf{A} 's term renaming and replacement of attributes referring to opaque and conditionally released identities with their respective values. The parameter $\mathcal{P}_{\mathbf{A}}$ of the function is party \mathbf{A} 's data repository in its state at the time of having the authenticated channel with \mathbf{C}_i , that is, the time of establishing the identity.

Delegation. The case of establishing a delegation identity relationship requires further considerations in addition. The relations between the identifier objects required for establishing an identity relationship with the pseudonyms has been discussed further above as part of the prerequisites. The subject term of the new identity as specified through the identifier object $\mathbf{t}_{p'}$ of ϕ refers to the same party, the delegater \mathbf{D} , to which the subject term \mathbf{D} uses for referring to itself in the statement it proves to the delegation certifier \mathbf{C}_{k+1} for initiating the delegation refers. Those subject terms used for referring to \mathbf{D} are different, though.

The semantics of those parties being the same gets established only through \mathbf{A} providing the delegation token \mathcal{V} it has received from \mathbf{D} , comprising also the pseudonym \mathbf{D} has used towards \mathbf{C}_{k+1} , and requesting the delegation certificate to be issued for the subject of identifier object $\mathbf{t}_{p'}$, also referring to \mathbf{D} . Until that, the latter identifier object could have referred to any party as its subject in case it is a newly generated delegation registration identifier object.

Through issuing of the delegation certificate, the registration identifier object \mathbf{A} has established for subject \mathbf{D} and the corresponding registration domain get cryptographically related to party \mathbf{D} and \mathbf{A} receives the capability of authenticating a statement about \mathbf{D} decorated with a security symbol towards other parties using the private certificate obtained in the instance of the **EstIdtyRel** protocol.

General. As a minimum requirement, the authentication property gets transferred from a channel between \mathbf{A} and a certifier to a channel between \mathbf{A} and a relying party in a data release protocol. In the delegation case, the authentication property gets transferred from a channel between the delegater \mathbf{D} and the delegation certifier to a channel between \mathbf{A} and \mathbf{B} . This means, the new statement is ensured to be about the same party. This transfer of authentications is realized cryptographically by issuing the new credential on the same private key the identifier object \mathbf{t}_p is based on, and, for the delegation case, also on the delegation private key the delegation registration identifier object $\mathbf{t}_{p'}$ is based on.

The formula $\tilde{\phi}$ of the formula ϕ proven as a prerequisite to the protocol **EstIdtyRel** may optionally comprise assertions about a registration identity of \mathbf{A} in order to ensure \mathbf{B} that \mathbf{A} has used a proper registration identifier, i.e., is a registered party in the system. Only if this is ensured, the authentication of the party with the certifier may be assumed to hold and thus can be transferred to another party through using a credential as explained in the model in Chapter 3, which is equivalent to transferring the security symbol annotating the channel endpoint between the channels. Requirements regarding the assertion to be proven are determined by the

authorization policy of **B**.

Note that the original extension [BCS12b, BCS12a] of the Maurer–Schmid channel model abstracts, for obtaining a simpler model, the complexity of the functions \mathcal{F} , \mathcal{F}^R , \mathcal{F}^D , and \mathcal{F}^{DR} and thus abstracts the corresponding aspects of the languages for expressing the formulae and how they relate.

In our secure channel model of Chapter 3, all formulae used for expressing rules refer to terms as observed by the recipient channel endpoint of the formula, that is, the party on the opposite side on the channel of the statement annotation. The corresponding formulae using terms of the source party are not represented and are a detail not relevant to the model, though, relevant for the logic-based data representation discussed in the current chapter and Chapter 4. This observation applies to all protocols discussed in the current chapter.

5.4 Release of Data

The release of certified data by a party **A** to a party **B** is done through the protocol **RelData**. An instance of this protocol comprises the release of a data statement and a cryptographic proof of correctness of the statement.

5.4.1 Protocol for Releasing Data

The protocol for releasing data is a 2-party protocol between a party **A**, the *prover* or *data provider*, and a party **B**, the *verifier* or *data recipient*. At the time of starting the protocol, **A** knows **B** at least under a pseudonymous identifier t_{sid_B} . This can be **B**'s public identifier (pseudonym) in case of **B** being a service provider acting under its public pseudonym, which is the common case. For the case of **B** not acting under a public pseudonym, it needs to establish a pseudonymous identifier t_{sid_B} with **A** for realizing a two-way pseudonymous interaction between the parties. Furthermore, we assume that **A** has established one or multiple pseudonyms with party **B** for the parties the statement it releases towards **B** is about, that is, itself and optionally one or multiple delegaters.⁴⁸

5.4.1.1 Protocol Interface

Next we present the interface of the protocol **RelData** in detail, starting with a high-level view thereof.

RelData protocol interface:

⁴⁸This assumption can be relaxed when extending the release protocol to support the establishment of identifier objects as mentioned previously.

Input	
A	$\phi_{\mathbf{A}}, \mathbf{t}_{sid_{\mathbf{B}}}, \{(\mathbf{o}_i, \{\nu_{l_{\mathbf{A}},i}\}_{1 \leq l_{\mathbf{A}},i \leq n_{\mathbf{A}},i})\}_{1 \leq i \leq n_{\mathbf{A}}^\diamond}$
B	—
Output	
A	$\phi'_{\mathbf{A}}, \{(\mathbf{o}'_j, \{\nu'_{l'_{\mathbf{A}},j}\}_{1 \leq l'_{\mathbf{A}},j \leq n'_{\mathbf{A}},j})\}_{1 \leq j \leq \hat{n}_{\mathbf{A}}^\diamond}, success_{\mathbf{A}}$
B	$\phi'_{\mathbf{B}}, \{(\mathbf{o}'_k, \{\nu'_{l'_{\mathbf{B}},k}\}_{1 \leq l'_{\mathbf{B}},k \leq n'_{\mathbf{B}},k})\}_{1 \leq k \leq \hat{n}_{\mathbf{B}}^\diamond}, success_{\mathbf{B}}$

The *input* comprises a data statement $\phi_{\mathbf{A}}$, a subject identifier $\mathbf{t}_{sid_{\mathbf{B}}}$ **B** is known to **A** under, and a set of metadata predicates ν_{\cdot} associated with the existing or to-be-generated objects \mathbf{o}_i referred to from within $\phi_{\mathbf{A}}$, that is, identifier objects, identities, conditionally released identities, and opaque identities, as an input of party **A**. Party **B** does not provide input explicitly passed at the API level for this protocol because **A** fully determines the statement to be released to **B**. Cryptographic key material need not be passed as input to the protocol because it can be obtained during the execution of the protocol [Som11].

The *output* of **A** comprises a new data statement $\phi'_{\mathbf{A}}$ based on $\phi_{\mathbf{A}}$ as well as a set of metadata predicates on the objects that are referred to in $\phi'_{\mathbf{A}}$ returned as output on the side of **A**. **B**'s output comprises, analogous to **A**'s, a data statement $\phi'_{\mathbf{B}}$ based on $\phi_{\mathbf{A}}$ and metadata predicates on the objects referred to in $\phi'_{\mathbf{B}}$ returned as output on the side of party **B**. The output of both parties comprises a boolean flag *success* indicating successful execution of the protocol.

The objects and associated metadata entities being input and output of the protocol are detailed next. The terms used for referring to the objects in **A**'s and **B**'s output in the interface specification are the ones used in **A**'s input formula after a renaming for obtaining the output—details hereto are provided further below.

Cryptographic entities are input and output through metadata predicates associated with terms representing the objects the entities correspond to. Such metadata predicate has the form as explained in (5.2), e.g., $\diamond(\mathbf{o}_i, \text{“pubCryptoMaterial”}, \xi^\diamond)$, with ξ^\diamond being cryptographic material as indicated through the “pubCryptoMaterial” constant, expressing that the metadata is the corresponding public, that is, shared, cryptographic material for object \mathbf{o}_i . For the objects to be generated in the protocol, that is, objects, the corresponding cryptographic entities of which are generated, the input may still comprise key material or other metadata.

The generated objects that are output of the protocol can be referred to in future protocols for relating their attributes to other attributes without revealing them. This requires that the cryptographic metadata being output as representative entities of the objects be input to the future protocols.

Metadata input. The identifier objects $\mathbf{t}_1^{\text{idf}}, \dots, \mathbf{t}_{\hat{\ell}^{\text{idf}}}^{\text{idf}}$ represent established pseudonyms **A** is the holder of about **A** or third parties. An object $\mathbf{t}_i^{\text{idf}}, 1 \leq i \leq \hat{\ell}^{\text{idf}}$ is associated with metadata predicates comprising its underlying cryptographic objects used in the cryptographic protocol: the private part $\tilde{\kappa}_i^{\text{gc}}$ and the public part $\hat{\kappa}_i^{\text{gc}}$. The extension of to-be-generated identifier objects $\mathbf{t}_{\hat{\ell}^{\text{idf}}+1}^{\text{idf}}, \dots, \mathbf{t}_{\hat{\ell}^{\text{idf}}}^{\text{idf}}$ of the RelData protocol represents those identifier objects through a term each without the need of

associating metadata with them in the input.

Note that the private key or keys of party **A** are not made explicit in the protocol interface as they are modeled to be part of cryptographic objects $\tilde{\kappa}_i^{\text{gc}}$.

The identities $\mathbf{t}_1^{\text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\text{id}}$ correspond to private certificates the party holds as part of identity relationships or to dummy certificates for simulated parts of the cryptographic proof to be computed. Let $\mathbf{t}_1^{\text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\text{id}}$ correspond to identities for which the party has identity relationships and thus private certificates, that is, their specification formulae being elements of the set $\hat{\Phi}^{\text{id}}$ that is part of the output of the result of an instance of the matching algorithm of Sec. 5.4.6. Let furthermore the remaining identities referred to through $\mathbf{t}_{\ell^{\text{id}}+1}^{\text{id}}, \dots, \mathbf{t}_{\ell^{\text{id}}}^{\text{id}}$ be dummy identities. An identity $\mathbf{t}_i^{\text{id}}, 1 \leq i \leq \ell^{\text{id}}$ is associated with its underlying certificate $\tilde{\kappa}_i^{\text{cr}}$ through a metadata predicate. A dummy identity $\mathbf{t}_i^{\text{id}}, \ell^{\text{id}} + 1 \leq i \leq \ell^{\text{id}}$ is associated with a dummy certificate χ_i^{cr} . A public key PK_i^{cr} is further metadata required as input for an identity $\mathbf{t}_i^{\text{id}}, 1 \leq i \leq \ell^{\text{id}}$ and is obtained within the protocol.⁴⁹ The predicates for the cryptographic keys are decorated with $\langle\langle \rangle\rangle$ symbols in the interface specification for indicating that they may be retrieved from within the protocol. For the non-dummy identities, a private key SK_i^{cr} is required in the protocol and obtained through the identifier relationship. The dummy certificates χ_i^{cr} can be created during the cryptographic protocol by obtaining the certificate structure using the certifier metadata associated with the corresponding identities.

The opaque identities $\mathbf{t}_1^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ represent hidden attributes generated in previous executions of instances of the **RelData** protocol that can be related to attributes of other objects in the current protocol. An opaque identity $\mathbf{t}_i^{\text{oid}}, 1 \leq i \leq \ell^{\text{oid}}$ is associated with a cryptographic commitment $\hat{\kappa}_i^{\text{dfc}}$ shared between the parties and the opening information $\tilde{\kappa}_i^{\text{dfc}}$, comprising the constituting attribute tuples, private to **A**. The to-be-generated opaque identities $\mathbf{t}_{\ell^{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ do not require to be associated with cryptographic metadata as an element $\hat{\kappa}_i^{\text{dfc}}$ contains the cryptographic key material required for the protocol, which is derived during the computation of the protocol instance in which it has been generated. For each $\mathbf{t}_i^{\text{oid}}$ with $\ell^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}$, private input χ_i^{dfc} needs to be input by **A**. An entity χ_i^{dfc} is generated when finding or otherwise defining the formula $\phi_{\mathbf{A}}$ to be released, e.g., through policy matching. It comprises the attribute tuples of the function $\mathcal{X}_{\mathbf{t}_i^{\text{oid}}}^{\text{oid}}$ corresponding to χ_i^{dfc} according to the semantics.

Technically, $\hat{\kappa}_i^{\text{dfc}}$ is a Damgård–Fujisaki–Okamoto commitment [DF02] and $\tilde{\kappa}_i^{\text{dfc}}$ is the corresponding opening information. Thereby, $\tilde{\kappa}_i^{\text{dfc}}$ is the technical encoding, as defined through the structure of $\mathbf{t}_i^{\text{oid}}$, of the tuples of the function $\mathcal{X}_{\mathbf{t}_i^{\text{oid}}}^{\text{oid}}$ corresponding to the opaque identity according to the semantics of Sec. 4.11 and an additional

⁴⁹Note that public and private keys are indexed as their corresponding certificates, although multiple certificates may be issued using the same public key and on the same private key. In the basic case of the cryptographic protocols, a single private key per user is employed, unless the user has multiple registrations in the system. In an implementation, safeguards must ensure that the private key be not exposed, e.g., by passing it as a reference to a key on a tamper-resistant security token.

randomizing value for achieving the statistical hiding property of the cryptographic commitment.

The conditionally released identities $\mathbf{t}_1^{\text{crd}}, \dots, \mathbf{t}_{\ell^{\text{crd}}}^{\text{crd}}$ represent conditionally released identities that have been generated in previous instances of the RelData protocol. A $\mathbf{t}_i^{\text{crd}}, 1 \leq i \leq \hat{\ell}^{\text{crd}}$ is associated with the entity $\tilde{\kappa}_i^{\text{enc}}$ private to **A** representing the technical encoding of the attributes of $\mathbf{t}_i^{\text{crd}}$ and $\tilde{\kappa}_i^{\text{enc}}$ being the shared ciphertext. The to-be-generated conditionally released identities $\mathbf{t}_{\ell^{\text{crd}}+1}^{\text{crd}}, \dots, \mathbf{t}_{\ell^{\text{crd}}}^{\text{crd}}$ represent ciphertexts and corresponding plaintext entities to be generated within the protocol execution and being protocol output. Every $\mathbf{t}_j^{\text{crd}}, 1 \leq j \leq \ell^{\text{crd}}$ is associated with a public key PK_j^{enc} for encryption, the corresponding predicate being annotated with $\langle\langle \rangle\rangle$. For each $\mathbf{t}_i^{\text{crd}}$ with $\hat{\ell}^{\text{crd}} + 1 \leq i \leq \ell^{\text{crd}}$, private input χ_i^{enc} needs to be input by **A**, analogous to χ_i^{dfc} above. Each χ_i^{enc} is generated when determining $\phi_{\mathbf{B}}$ and comprises the attribute tuples of the function $\mathcal{X}_{\mathbf{t}_i^{\text{crd}}}^{\text{crd}}$ corresponding to $\mathbf{t}_i^{\text{crd}}$ following the formal semantics.

Technically, $\tilde{\kappa}_i^{\text{enc}}$ is a ciphertext computed through the Camenisch–Shoup verifiable encryption scheme [CS03] of the attribute tuples of $\tilde{\kappa}_i^{\text{enc}}$, where $\tilde{\kappa}_i^{\text{enc}}$ is the technical encoding, as governed through the structure for $\tilde{\kappa}_i^{\text{enc}}$, of the attribute tuples, except for the *condition* and *recipient* attributes, of the set-based specification of the function $\mathcal{X}_{\mathbf{t}_i^{\text{crd}}}^{\text{crd}}$ corresponding to the conditionally released identity in the semantics defined in Sec. 4.11.

Note that both $\chi_i^{\text{dfc}}, \hat{\ell}^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}$ and $\chi_i^{\text{enc}}, \hat{\ell}^{\text{crd}} + 1 \leq i \leq \ell^{\text{crd}}$ above could be easily computed as part of the RelData protocol from $\phi_{\mathbf{A}}$ and its $\langle \rangle$ annotations, though, they are already computed as part of the policy matching process and thus available to the protocol. Also note that those elements are not $\langle\langle \rangle\rangle$ -annotated as they are not available in **A**'s repository and thus cannot be retrieved by the protocol implementation.

We summarize the discussed input metadata predicates next, with objects referred to through the terms **A** uses in its local representation. The tabular notation indicates for each metadata predicate through the party names in the rightmost columns which parties receive them as input.

RelData Metadata input:

$\langle\langle \diamond(\mathbf{t}_i^{\text{idf}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{gc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{idf}}$	A	B
$\langle\langle \diamond(\mathbf{t}_i^{\text{idf}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{gc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{idf}}$	A	
$\langle\langle \diamond(\mathbf{t}_i^{\text{idv}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{crt}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{idv}}$	A	
$\langle\langle \diamond(\mathbf{t}_i^{\text{idv}}, \text{"privCryptoMaterial"}, \chi_i^{\text{crt}}) \rangle\rangle$ for $\hat{\ell}^{\text{idv}} + 1 \leq i \leq \ell^{\text{idv}}$	A	
$\langle\langle \diamond(\mathbf{t}_i^{\text{idv}}, \text{"pubKey"}, \text{PK}_i^{\text{crt}}) \rangle\rangle$ for $1 \leq i \leq \ell^{\text{idv}}$	A	B
$\langle\langle \diamond(\mathbf{t}_i^{\text{oid}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{dfc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{oid}}$	A	B
$\langle\langle \diamond(\mathbf{t}_i^{\text{oid}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{dfc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{oid}}$	A	
$\diamond(\mathbf{t}_i^{\text{oid}}, \text{"privCryptoMaterial"}, \chi_i^{\text{dfc}})$ for $\hat{\ell}^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}$	A	
$\langle\langle \diamond(\mathbf{t}_i^{\text{crd}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{enc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{crd}}$	A	B
$\langle\langle \diamond(\mathbf{t}_i^{\text{crd}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{enc}}) \rangle\rangle$ for $1 \leq i \leq \hat{\ell}^{\text{crd}}$	A	
$\diamond(\mathbf{t}_i^{\text{crd}}, \text{"privCryptoMaterial"}, \chi_i^{\text{enc}})$ for $\hat{\ell}^{\text{crd}} + 1 \leq i \leq \ell^{\text{crd}}$	A	
$\langle\langle \diamond(\mathbf{t}_i^{\text{crd}}, \text{"pubKey"}, \text{PK}_i^{\text{enc}}) \rangle\rangle$ for $1 \leq i \leq \ell^{\text{crd}}$	A	B

Note that the $\langle\langle \rangle\rangle$ -annotated entities can be obtained by the implementation of the protocol and do not need to be passed as input explicitly at the API level. Retrieval of those entities is done based on the object term used in $\pi^1(\diamond_j)$ of the metadata predicate \diamond_j that refers to the object the predicate applies to in the party's local representation of data. We thereby assume that the protocol implementation has access to the data repository \mathcal{P} of the party. We next discuss this for protocol participants **A** and **B**, using formula $\phi_{\mathbf{A}}$ or $\phi'_{\mathbf{B}}$, respectively. **B** only requires a subset of the inputs of **A**, namely the non-private materials.

$\widehat{\kappa}_i^{\text{gc}}$ and $\widetilde{\kappa}_i^{\text{gc}}$ for $1 \leq i \leq \widehat{\ell}^{\text{idf}}$ and $\widehat{\kappa}_i^{\text{crt}}$ for $1 \leq i \leq \widehat{\ell}^{\text{idv}}$ are retrieved from the corresponding identifier and identity relationships which **A** is holder of by **A**, and $\widehat{\kappa}_i^{\text{gc}}$ for $1 \leq i \leq \widehat{\ell}^{\text{idf}}$ from identifier relationships another party has established with **B** by **B**. Public keys PK_i^{crt} related to identities are retrieved from the profile data entries in \mathcal{P} representing the public key certificates of the keys, using the certifier specification sub-formulae of $\phi_{\mathbf{A}}$ or $\phi'_{\mathbf{B}}$.

$\widehat{\kappa}_i^{\text{dfc}}$ and $\widetilde{\kappa}_i^{\text{dfc}}$ for $1 \leq i \leq \widehat{\ell}^{\text{oid}}$ are retrieved by **A** from the metadata predicates related to the data tracking entries for the previously released formulae by **A**, where those cryptographic entities have been created in the protocols for proving those formulae. $\widehat{\kappa}_i^{\text{dfc}}$, $1 \leq i \leq \widehat{\ell}^{\text{oid}}$ are retrieved by **B** from profile data entries created for those formulae after previous protocol executions through which it received data through the RelData protocol. Analogously, $\widehat{\kappa}_i^{\text{enc}}$ and $\widetilde{\kappa}_i^{\text{enc}}$ for $1 \leq i \leq \widehat{\ell}^{\text{crid}}$ are retrieved by **A** from the data track entries for the formulae for the release of which those entities have been generated and $\widehat{\kappa}_i^{\text{enc}}$, $1 \leq i \leq \widehat{\ell}^{\text{crid}}$ and from corresponding profile data entries by **B**. The encryption keys PK_i^{enc} , $1 \leq i \leq \widehat{\ell}^{\text{crid}}$ are obtained by both **A** and **B** through executing appropriate protocols, e.g., PKI-based protocols, from parties as determined through the third-party specification formulae corresponding to the conditional data recipients under whose keys to perform the encryption.

Metadata output. For the to-be-generated identifier objects $\mathbf{t}_{\widehat{\ell}^{\text{idf}}+1}^{\text{idf}}, \dots, \mathbf{t}_{\widehat{\ell}^{\text{idf}}}^{\text{idf}}$, the metadata output for an object $\mathbf{t}_i^{\text{idf}}$, $\widehat{\ell}^{\text{idf}}+1 \leq i \leq \widehat{\ell}^{\text{idf}}$ comprises the private part of the established cryptographic pseudonym $\widetilde{\kappa}_i^{\text{gc}}$ and the public part $\widehat{\kappa}_i^{\text{gc}}$ for **A** and the public part $\widehat{\kappa}_i^{\text{gc}}$ for **B**.⁵⁰

There is no output corresponding to private certificates—the cryptographic material resulting in the protocol for a certificate $\widetilde{\kappa}_i^{\text{crt}}$ is a proof of knowledge $\widetilde{\kappa}_i^{\text{crt}}$ of a randomized certificate $\widetilde{\kappa}_i^{\text{crt}}$ based on $\widetilde{\kappa}_i^{\text{crt}}$ showing the correctness of (parts of) the formula $\phi'_{\mathbf{B}}$, though, not being reused in a future protocol. Note that a credential system supporting privacy-preserving revocation of credentials, e.g., using the approach of Camenisch and Lysyanskaya [CL02a] based on dynamic accumulators requires that a cryptographic credential $\widetilde{\kappa}_i^{\text{crt}}$ corresponding to an identity $\mathbf{t}_i^{\text{idv}}$ with $1 \leq i \leq \widehat{\ell}^{\text{idv}}$ be updated with respect to the revocation feature before being used in a RelData protocol. Otherwise, the generated cryptographic proof may not verify at **B**'s side. Such credential update is assumed to have been executed before the RelData protocol is invoked. In terms of architecture, this can be done periodically

⁵⁰The functionality of creating identifier objects within the RelData protocol is an extension.

by a component related to the cryptographic protocols of the architecture, an invocation of a pre-protocol to RelData performing the update, or a combination thereof. It is crucial that the complexity of this be encapsulated into a component related to the cryptographic protocols and abstracted from the programming interface.

The to-be-generated opaque identities $\mathbf{t}_{\hat{\ell}^{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ have cryptographic metadata output: An opaque identity $\mathbf{t}_i^{\text{oid}}$ has commitment opening information $\tilde{\kappa}_i^{\text{dfc}}$ and commitment $\hat{\kappa}_i^{\text{dfc}}$ for **A** as metadata output and only the commitment $\hat{\kappa}_i^{\text{dfc}}$ for **B**.

The protocol computes an integer attribute tuple from χ_i^{dfc} through a representation of the attributes of the latter following the object specification structure for $\tilde{\kappa}_i^{\text{dfc}}$. The structure is canonically determined through the attribute types in χ_i^{dfc} and formula $\phi_{\mathbf{A}}$. The commitment $\hat{\kappa}_i^{\text{dfc}}$ is computed as a Damgård–Fujisaki–Okamoto commitment over the integer attribute tuple, thereby creating an additional randomization integer. The attribute tuple and the additional integer form the integer tuple $\tilde{\kappa}_i^{\text{dfc}}$.

Similarly, the to-be-generated conditionally released identities $\mathbf{t}_{\hat{\ell}^{\text{crid}}+1}^{\text{crid}}, \dots, \mathbf{t}_{\ell^{\text{crid}}}^{\text{crid}}$ have output as follows: A $\mathbf{t}_i^{\text{crid}}$ has its protocol-generated private attribute tuples $\tilde{\kappa}_i^{\text{enc}}$ and the shared ciphertext $\hat{\kappa}_i^{\text{enc}}$ as output for **A** and $\hat{\kappa}_i^{\text{enc}}$ for **B**.

In the protocol, $\tilde{\kappa}_i^{\text{enc}}$ is derived from χ_i^{enc} analogous as above through a representation of the attributes of the latter following the object specification structure for $\tilde{\kappa}_i^{\text{enc}}$. The structure is derived analogous as for a $\tilde{\kappa}_i^{\text{dfc}}$. The ciphertext $\hat{\kappa}_i^{\text{enc}}$ is obtained through Camenisch–Shoup verifiable encryption of the tuple $\tilde{\kappa}_i^{\text{enc}}$ under the condition and the public key specified for $\mathbf{t}_i^{\text{crid}}$ through $\phi_{\mathbf{A}}$.

Note that the cryptographic entities $\tilde{\kappa}_i^{\text{crt}}$, $\hat{\kappa}_i^{\text{enc}}$, and $\hat{\kappa}_i^{\text{dfc}}$ corresponding to identities, conditionally released identities, and opaque identities, respectively, comprise structural information in the form of cryptographic object structures—a certificate structure for a certificate—specifying aspects such as available attributes and their encoding.

The metadata output predicates for both parties are summarized next, using the same notation as for the metadata input.

RelData Metadata output:

$\diamond(\mathbf{t}_i^{\text{idf}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{gc}})$ for $\hat{\ell}^{\text{idf}} + 1 \leq i \leq \ell^{\text{idf}}$	A	B
$\diamond(\mathbf{t}_i^{\text{idf}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{gc}})$ for $\hat{\ell}^{\text{idf}} + 1 \leq i \leq \ell^{\text{idf}}$	A	
$\diamond(\mathbf{t}_i^{\text{oid}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{dfc}})$ for $\hat{\ell}^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}$	A	B
$\diamond(\mathbf{t}_i^{\text{oid}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{dfc}})$ for $\hat{\ell}^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}$	A	
$\diamond(\mathbf{t}_i^{\text{crid}}, \text{"pubCryptoMaterial"}, \tilde{\kappa}_i^{\text{enc}})$ for $\hat{\ell}^{\text{crid}} + 1 \leq i \leq \ell^{\text{crid}}$	A	B
$\diamond(\mathbf{t}_i^{\text{crid}}, \text{"privCryptoMaterial"}, \tilde{\kappa}_i^{\text{enc}})$ for $\hat{\ell}^{\text{crid}} + 1 \leq i \leq \ell^{\text{crid}}$	A	

5.4.1.2 Formulae

The input and output formulae of the RelData protocol are based on our formal data representation of Chapter 4. The formulae are expressed in the language $\mathcal{L}^{\text{p/c}}$, that is, $\phi_{\mathbf{A}}, \phi'_{\mathbf{A}}, \phi'_{\mathbf{B}} \in \mathcal{L}^{\text{p/c}}$.

The syntax of $\mathcal{L}^{\text{p/c}}$ applies to our credential-based cryptographic protocols, other

protocols, e.g., such based on conventional certificates or an online certifier, may have a less or more expressive syntax, respectively. In any case, the specification language for any practical technology is a fragment of $\mathcal{L}^{p/g}$.

The meaning of the formulae in the RelData protocol is different to the meaning of formulae in the protocols explained earlier: The formulae are statement-type formulae and specify the data statement to be released by **A** to **B**, proven correct with respect to cryptographic material corresponding to the referred objects.

The input formula $\phi_{\mathbf{A}}$ is constructed based on formulae used in **A**'s local representation of identifier and identity relationships and the constant terms used in those, while the output formulae $\phi'_{\mathbf{A}}$ and $\phi'_{\mathbf{B}}$ express exactly the same semantics as $\phi_{\mathbf{A}}$, though using different terms for referring to certain objects and parties as explained later. $\phi_{\mathbf{A}}$ refers to objects that party **A** holds using the terms party **A** uses in its local data representation in \mathcal{P} —see Sec. 4.8.

We recall that a formula of $\mathcal{L}^{p/c}$ comprises multiple sub-formulae in the preamble and one sub-formula for specifying the attribute assertion. For the details we refer the reader to the discussions in Sec. 4.5. The cryptographic reason behind only allowing the \wedge -connective in sub-formulae for identity introduction and conditionally released identity introduction is that—because of how the cryptographic protocols operate—only a single public key can be associated with an object throughout a formula and thus the certifier or conditional data recipient needs to be specified in the top-level \wedge -node of the formula. For Camenisch–Shoup verifiable encryption [CS03], only one release condition can be specified for a conditionally released identity per formula, thus it needs to be associated with this conditionally released identity in the top-level \wedge -node.⁵¹

The identifier objects $t_1^{\text{idf}}, \dots, t_{\ell_{\text{idf}}}^{\text{idf}}$ referred to may be any of the identifier objects from identifier relationships **A** has with **B** about itself or third parties, which means, those which have the term $t_{\text{sid}_{\mathbf{B}}}$ as identifier in a metadata predicate indicating the party they have been established with.

5.4.2 Term Derivation and Renaming

The terms used in the output formulae $\phi'_{\mathbf{A}}$ and $\phi'_{\mathbf{B}}$ and in the metadata are computed from the terms of the input $\phi_{\mathbf{A}}$ through *term renaming*. This renaming assigns constant terms to protocol-generated objects and ensures that no undesired linkabilities are established through inappropriate re-use of terms throughout multiple protocol instances. The renaming explained in the current section is performed to obtain formulae $\phi'_{\mathbf{A}}$ and $\phi'_{\mathbf{B}}$ from $\phi_{\mathbf{A}}$. Note that the renaming of objects the names of which are protocol dependent can only be done once the cryptographic protocol steps of **A** have been computed.

⁵¹For the Camenisch–Damgård verifiable encryption scheme [CD00], this could be relaxed and different release conditions could be specified in different disjunctive branches of the formula.

5.4.2.1 Term Derivation

Terms referring to objects in the input formulae and metadata that are yet to be created within the protocol execution need to be replaced with concrete constant terms in the output based on the executed cryptographic protocol. This holds for protocol-generated conditionally released identities referred to through $\mathbf{t}_{\ell_{\text{crd}}+1}^{\text{crd}}, \dots, \mathbf{t}_{\ell_{\text{crd}}}^{\text{crd}}$ and protocol-generated opaque identities $\mathbf{t}_{\ell_{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell_{\text{oid}}}^{\text{oid}}$ —both of which can be created as part of the **RelData** protocol. One can—for reducing the number of message exchanges over the network—as an extension also have identifier objects created as part of the protocol instead of in a separate protocol.

For a conditionally released identity $\mathbf{t}_i^{\text{crd}}, \ell_{\text{crd}} + 1 \leq i \leq \ell_{\text{crd}}$, let $\hat{\kappa}_i^{\text{enc}}$ be the underlying shared cryptographic object, that is, a ciphertext based on the Camenisch–Shoup verifiable encryption scheme [CS03] for our protocols. The term for referring to $\mathbf{t}_i^{\text{crd}}$ in the output is computed as $\mathbf{t}_i^{\text{crd}} := \eta(\zeta_o(\hat{\kappa}_i^{\text{enc}}), \text{crd})$ as part of the protocol.

For an opaque identity $\mathbf{t}_i^{\text{oid}}, \ell_{\text{oid}} + 1 \leq i \leq \ell_{\text{oid}}$, let $\hat{\kappa}_i^{\text{dfc}}$ be the underlying shared object available to both parties. The term for referring to this opaque identity in the output is computed as $\mathbf{t}_i^{\text{oid}} := \eta(\zeta_o(\hat{\kappa}_i^{\text{dfc}}), \text{oid})$ as part of the protocol.

For an identity $\mathbf{t}_i^{\text{idy}}, 1 \leq i \leq \ell_{\text{idy}}$, let $\tilde{\kappa}_i^{\text{crt}}$ either be the randomized private certificate (signature) holdership of which is proven or a tuple sampled from a probability distribution indistinguishable from that of the signature tuples for the corresponding public key a zero-knowledge proof of knowledge protocol simulation transcript over which is generated. This results in a cryptographic zero-knowledge proof transcript or a protocol simulation transcript, respectively, denoted as $\bar{\kappa}_i^{\text{crt}}$. The output term for $\mathbf{t}_i^{\text{idy}}$ is computed as $\mathbf{t}_i^{\text{idy}} := \eta(\zeta_o(\bar{\kappa}_i^{\text{crt}}), \text{id})$ as part of the protocol. Note that unlike for identifier objects which are never renamed, an identity based on a private certificate or an identity a proof of which is to be simulated needs to be renamed to not establish unintended linkability with previous protocol instances or reveal information on which identities are backed by private certificates and for which a proof simulation is computed in the cryptographic proof.

The above-discussed derivation and renaming of terms is captured in a function ϱ^{prf} for the prove protocol instance and added as a metadata predicate in **A**’s output, to be used for precisely tracking the released data, including terms used, in the data track. This can facilitate a presentation of released data to the user, while comprising information on linkability of protocols.

5.4.2.2 Term Renaming

The terms used for referring to values of the *holder* and *subject* attributes of objects are renamed as part of the protocol. The idea hereto is that the terms used for referring to the *holder* and *subject* of objects are those terms used for the holder and subject of identifier objects. Complications are incurred through multiple identifier objects with different holder and subject terms that may refer to the same party, due to the approach following which those terms are computed for each identifier object. See Sec. 4.8 for details on those aspects.

We next explain the term renaming for a formula for the prominent special case of the general renaming regime introduced in Sec. 4.8 of the formula referring to one identifier object with \mathbf{A} as subject and one with a delegater \mathbf{D} as subject as the identifier objects referred to in the identifier introduction sub-formula. Let \mathbf{p}_1 represent an identifier object for an identifier of and about \mathbf{A} with \mathbf{B} and \mathbf{p}_2 an identifier object of \mathbf{A} about a delegater \mathbf{D} with \mathbf{B} . Let the terms $\bar{\mathbf{t}}_{h,\mathbf{A},1} = \bar{\mathbf{t}}_{s,\mathbf{A},1}$ be used in \mathbf{A} 's input formula to refer to the holder and subject of the identifier object \mathbf{p}_1 . Let furthermore the term $\bar{\mathbf{t}}_{h,\mathbf{A},2}$ be the holder term and $\bar{\mathbf{t}}_{s,\mathbf{A},2}$ be the subject term of the identifier object \mathbf{p}_2 as used in \mathbf{A} 's formula. Note that the terms $\bar{\mathbf{t}}_{h,\mathbf{A},1}$, $\bar{\mathbf{t}}_{s,\mathbf{A},1}$, and $\bar{\mathbf{t}}_{h,\mathbf{A},2}$ are all equal following the local representation of identity data which $\phi_{\mathbf{A}}$ builds upon through construction. Let the term $\mathbf{t}_{h,\mathbf{A},1} := \varrho_{\mathbf{p}_1}^{\text{idf}}(\bar{\mathbf{t}}_{h,\mathbf{A},1})$ be the holder and $\mathbf{t}_{s,\mathbf{A},1} := \varrho_{\mathbf{p}_1}^{\text{idf}}(\bar{\mathbf{t}}_{s,\mathbf{A},1})$ the subject of \mathbf{p}_1 as determined through its term renaming function $\varrho_{\mathbf{p}_1}^{\text{idf}}$. Analogously, let $\mathbf{t}_{h,\mathbf{A},2} := \varrho_{\mathbf{p}_2}^{\text{idf}}(\bar{\mathbf{t}}_{h,\mathbf{A},2})$ be the holder and $\mathbf{t}_{s,\mathbf{A},2} := \varrho_{\mathbf{p}_2}^{\text{idf}}(\bar{\mathbf{t}}_{s,\mathbf{A},2})$ be the subject of the identifier object \mathbf{p}_2 according to its term renaming function $\varrho_{\mathbf{p}_2}^{\text{idf}}$. Recall that the renaming function of an identifier object captures the mapping of locally used terms in \mathbf{A} 's data representation to the terms used in interactions with the party or parties the identifier object has been established with.⁵² The functions $\varrho_{\mathbf{p}_1}^{\text{idf}}$ and $\varrho_{\mathbf{p}_2}^{\text{idf}}$ are retrieved by \mathbf{A} as metadata from its portfolio. Let the terms $\mathbf{t}_{h,\mathbf{A},1}$, $\mathbf{t}_{s,\mathbf{A},1}$ and $\mathbf{t}_{h,\mathbf{A},2}$ refer to the same party \mathbf{A} and $\mathbf{t}_{s,\mathbf{A},2}$ to a delegater \mathbf{D} .

The renaming proceeds as follows for the case presented above: Choose the identifier object \mathbf{p}_1 as the one for determining the term $\mathbf{t}_{h,\mathbf{A},1}$ which is the same term as $\mathbf{t}_{s,\mathbf{A},1}$ used for referring to \mathbf{A} in the output formula $\phi'_{\mathbf{B}}$. Choose furthermore the term $\mathbf{t}_{s,\mathbf{A},2}$ for referring to \mathbf{D} as the other subject in the output $\phi'_{\mathbf{B}}$. Replace each occurrence of term $\bar{\mathbf{t}}_{h,\mathbf{A},1}$ throughout $\phi_{\mathbf{A}}$ with $\mathbf{t}_{h,\mathbf{A},1}$, except for in logical equality predicates relating attributes of identifier objects other than \mathbf{p}_1 with constant terms. Replace the term $\bar{\mathbf{t}}_{s,\mathbf{A},2}$ with $\mathbf{t}_{s,\mathbf{A},2}$ throughout $\phi_{\mathbf{A}}$, except for in predicates relating attributes of identifier objects other than \mathbf{p}_2 with constant terms. Replace the term $\bar{\mathbf{t}}_{h,\mathbf{A},2}$ with $\mathbf{t}_{h,\mathbf{A},2}$, except for in predicates relating attributes of identifier objects other than \mathbf{p}_2 with constant terms. Introduce a new predicate $\mathbf{t}_{h,\mathbf{A},1} = \mathbf{t}_{h,\mathbf{A},2}$ using the logic's object equality, connected with \wedge to the predicates in the identifier object introduction sub-formula, for specifying the sameness of the renamed terms with the original terms because this semantics of the formula would otherwise be lost through the renaming.⁵³

This approach generalizes to an arbitrary number of identifier objects about parties being referred to in $\phi_{\mathbf{A}}$, particularly also multiple identifier objects about the same party. The basic idea in the generalized setting is to choose one identifier object for each party referred to through the same term in the input $\phi_{\mathbf{A}}$ and use this for determining the term renaming for referring to this party. The renaming is done throughout the formula, but not for predicates referring to attributes of the

⁵²Note that the terms are always the same for a given identifier object, thereby following the concepts underlying a pseudonym.

⁵³Note that this goes beyond a pure term replacement, though is required for retaining the equality semantics between terms referring to parties after term renaming and for obtaining an output formula $\phi'_{\mathbf{B}}$ that is logically equivalent to $\phi_{\mathbf{A}}$.

other identifier objects than the chosen one. Equalities are established between the terms that were the same in the input and have been renamed to different terms in the output. A special—and very common—case is when only a single identifier object is used for making statements about \mathbf{A} .

The explained approach also captures the case of identifier objects and identities based on different registration identifier objects. This is the case when a user obtains different registration pseudonyms and therefore different private keys, implying multiple private key domains (see Sec. 4.8). Such different domains are reflected by using different terms in the input to refer to the party \mathbf{A} that may be related by the logic's equality symbol $=$ and cryptographic proof thereof through a domain binding credential. Due to conducting a separate renaming for each different input term referring to the same party, this results in different output terms and therefore handles this case correctly.

The processing renames the terms used by \mathbf{A} in its local data representation to the terms to be used for the instance of the release protocol and introduces equalities where required to retain the semantics of $\phi_{\mathbf{A}}$ in $\phi'_{\mathbf{B}}$. This seemingly complicated approach is necessary for our logic-based data representation to achieve that a holder of identifier and identity relationships can match data contained in such against data requests using derivations in the underlying logic, and particularly also considering data about third parties, while supporting delegation and party registration.

When using credential protocols for proving the integrity of the formula with respect to the referred-to identifier and identity relationships, the new terms for referring to objects are determined by the cryptographic protocol and cannot be arbitrarily chosen by the party. A term t_i^{id} gets replaced with a term $t'^{\text{id}} := \eta(\zeta_o(\kappa_i^{\text{rt}}, \text{id}), \text{id})$. A term referring to a conditionally released identity gets replaced with the term $t_i^{\text{crid}} := \eta(\zeta_o(\kappa_i^{\text{enc}}, \text{crid}), \text{crid})$ and one referring to an opaque identity with term $t_i^{\text{oid}} := \eta(\zeta_o(\kappa_i^{\text{dfe}}, \text{oid}), \text{oid})$ —details about the computation have been explained further above. This renaming is done once the cryptographic protocol has been executed.

This leads to terms that are unique with overwhelming probability and lead to a collision with already-existing terms only with negligible probability with appropriate choice of character lengths of terms and assuming a polynomial number, in a security parameter, of terms being created in the system. From a logic-based modeling perspective this is an ideal situation because the terms are provably correctly chosen which may not be the case when different or no deterministic schemes for deriving the terms from cryptographic elements of the proof are used.

Once the term renaming has been done, the resulting formula is used as input to the cryptographic protocol for generating a cryptographic proof for the correctness of the formula. The data recipient does not need to perform the term renaming as it already receives the formula with the renamed terms. The respective party uses input to the cryptographic protocol as discussed further above and retrieves the cryptographic materials by lookup based on object names of objects in the formula from its repository.

The cryptographic proof can, when being generated as a non-interactive proof

based on a Fiat-Shamir signature, be sent to the data recipient together with the formula ϕ'_A . Note that, once the cryptographic proof has been computed, a final renaming step on certain elements of the formula is carried out as explained above to properly name the proof-dependent elements.

More details on the cryptographic protocols are presented in the form of a summary of the cryptographic protocol subsystem in Chapter 6.

5.4.3 Post Processing

Once a protocol instance for releasing data has been successfully executed, the releasing party **A** stores its protocol output, that is, the released formula ϕ'_A as well as any associated metadata in its data track with itself as subject, while the recipient stores the received formula ϕ'_B and metadata in a profile data entry with the subject being the value of the attribute *subjectId* for the party the data statement is about, that is, **A** or a referred-to delegater **D**, or both.

The recipient **B**'s metadata comprise cryptographic material proving the correctness of the released formula as well as cryptographic objects related to the opaque and conditionally released identities. The latter allow a third party, the respective conditional data recipient, to obtain the attribute values of a conditionally released identity if the attached condition is fulfilled.

Uncovering the attribute values of a conditionally released identity t_i^{crd} , that is, obtaining the function $\mathcal{X}_{t_i^{crd}}^{crd}$, through the corresponding conditional data recipient is performed through a separate protocol involving **B** and the conditional recipient. Thereby, **B** requests a decryption of $\widehat{\kappa}_i^{enc}$ and passes the condition that it claims to hold. The conditional recipient checks whether the condition holds and decrypts the ciphertext $\widehat{\kappa}_i^{enc}$ with the condition and its private decryption key as input.

The prover **A**'s metadata comprise the same as the recipient's fulfilling the discussed conditions on term names, and additionally the term renaming function applied to its input formula for obtaining the output. Both parties store the cryptographic objects related to the protocol-generated conditionally released and opaque identities, where **A** stores both the shared and private objects related to those. Both parties store the object specification formulae related to the generated conditionally released and opaque identities. Those objects are required in future instances of the RelData protocol between **A** and **B**, the specification formulae of which refer to the corresponding conditionally released and opaque identities.

5.4.4 Architecture Aspects

An instance of a RelData protocol is triggered by the party **A** who intends to release data, e.g., a user **A** interacting with a service provider **B**. Within our architecture, the RelData protocol is typically invoked from within the policy-driven *negotiation protocol* discussed in earlier work [Som11] for mutual release of (certified) data between two interacting parties. Note that establishing a confidential and one-way authenticated channel using standard PKI technology can already establish attribute knowledge about the authenticated party exploitable in a negotiation protocol and

can be viewed as a data release protocol based on traditional X.509 server-side certificates.

Multiple technologies for releasing data. A data statement can make use of multiple identity relationships and thus require, in the general case, multiple underlying protocols based on different technologies for the technical exchange of the data, e.g., a credential system for the release of certified data and a protocol for the plaintext release of attribute data. The proposed logic-based representation of data allows for this as it is largely protocol independent. The release of such combined formula needs to be handled through an appropriate component in an implementation architecture. This can be done by splitting a formula to be released into multiple formulae and execute protocols for the release accordingly, e.g., a credential protocol and a plaintext release protocol to release a compound statement comprising both identity information certified by third parties and claimed by the party. A data statement can also build on multiple formulae of different identity relationships with mutually “compatible” protocols. In this case, identities of different identity relationships can be related to each other, e.g., their subjects can be expressed to be the same party without revealing a unique identifier. Currently, private certificate systems are the only technology supporting such proofs that involve multiple identity relationships, even with different certifiers, and even allow for relating unrevealed attributes of such to each other. For uncertified claims this can be trivially achieved, though based on honesty of the claiming party and thus not interesting from a protocol perspective. See Sommer [Som11] for architecture discussions related to supporting multiple technologies.

Theft and sharing prevention. When relating multiple identifier objects or identities through their holder and subject attributes and using credential-based protocols for proving the given formula correct, using the same holder or subject for two different objects means that the holder or subject party of those is the same. The proof thereby shows that the objects have the same underlying secret key, and optionally additionally delegation secret key. Such key equality is strong evidence of sameness of the party because a party can be disincentivized or prevented from sharing its master secret key, or protected from theft thereof, through different technical or organizational means or a combination thereof, e.g., containment of the private key on a hardware token or encryption of a valuable external secret of the user under the private key [CL01]. See Bichsel [Bic07] for a discussion of theft and sharing prevention and references to relevant literature. For identity relationships for delegated identities some concepts apply analogously to the delegation private key the party holds.

5.4.5 Relation to the Channel Model

Referring to the channel transformation rule Auth-Cred for transferring authentications of our model in Chapter 3, the protocol RelData realizes the part of the channel transformation rule where **A** authenticates a statement ϕ' to **B** which is

based on statements \mathbf{A} has authenticated to certifiers \mathbf{C}_i for $1 \leq i \leq k$ and for which \mathbf{A} has established identity relationships based on those authentications. The symbol θ with appropriate decorations as referred to in the sequent for relating ϕ' to the party's repository in rule **Auth-Cred** refers to the set of identifier object specification formulae of \mathbf{A} for the identifier objects it has established with \mathbf{B} . The symbol η refers to the object specification formulae for the already-established or to-be established conditionally released identities. Analogously, χ refers to the object specification formulae for the existing or to-be-created opaque identities. The object specification formulae contained in η and χ are required for the modeling of the channel transformation in order to be able to establish the $\vdash_{\mathcal{O}}$ -relation in the logic calculus.

For rule **AuthDele-Cred** for transferring authentications of channels while also modeling delegation, the above applies analogously, while additionally the authenticated statement can be used to construct the statement ϕ' authenticated to \mathbf{B} . The functions $\mathcal{F}^{\mathbf{D}}$ and $\mathcal{F}^{\mathbf{DR}}$ show some differences to the corresponding functions for non-delegation issuing in terms of handling the authenticated formula with \mathbf{C}_{k+1} for considering the delegation. The details of the identity relationships for both the non-delegation and delegation case and their correspondance to the channel model have been discussed in Sec. 5.3.

Note that the protocol **RelData** can consider an arbitrary number of delegation credentials for one delegater and make statements about an arbitrary number of delegaters in one formula, while the channel transformation rule is constrained to a single credential for a single delegater \mathbf{D} . A formula comprising assertions about multiple delegaters is not crucial for practical systems.

5.4.6 Construction of $\phi_{\mathbf{A}}$

A formula $\phi_{\mathbf{A}}$ used as the main input specification for an instance of the **RelData** protocol needs to be available to \mathbf{A} before triggering the data release protocol. The most prominent way of defining the formula is that \mathbf{A} has received a data request ψ within a negotiation protocol, with ψ being a formula in a fragment of $\mathcal{L}^{r/g}$ corresponding to a specific protocol, which it needs to fulfill. Then it generates $\phi_{\mathbf{A}}$ from the request ψ through a formal process, possibly with additional external input, e.g., by the user.

For our credential protocols, the fragment of $\mathcal{L}^{r/c}$ of $\mathcal{L}^{r/g}$ is used as request language, while for any other mechanism, an appropriate fragment of $\mathcal{L}^{r/g}$ can be employed. For each mechanism, a pair of request and statement languages is defined, for our credential protocols this is $(\mathcal{L}^{r/c}, \mathcal{L}^{p/c})$.

The language $\mathcal{L}^{r/c}$ has been defined in Sec. 4.6 based on the overall data model. Let us recall here that main differences to the statement language are that free variables are used for expressing request semantics—a free variable stands for any of the possible concrete instantiations of this variable in a statement formula provided as a response to a request. Furthermore, ontology-based generalizations can be used in the specification, e.g., of third parties.

The process of finding fulfilling formulae for a data request (policy) using the

party's repository \mathcal{P} is denoted as *matching* or *policy matching*. The matching process builds on the identifier specification formulae of the identifier relationships of the party, the identity specification formulae of the identity relationships, the profile data entries for third parties, and the data track entries for formulae during the proof protocol for which opaque and conditionally released identities have been computed.

5.4.6.1 Input

Let ψ_i be the formula specifying $\mathbf{t}_i^{\text{idf}}$ of identifier relationship ϵ_i^{idf} of the party and $\Phi^{\text{idf}} = \{\psi_i : 1 \leq i \leq \ell^{\mathcal{R}^{\text{idf}}}\}$ the set of the formulae of all its identifier relationships. Let $\psi'_{j,j'}$ be the formulae specifying the identity \mathbf{t}_j^{id} of identity relationship ϵ_j^{id} and $\Phi^{\text{id}} = \{\psi'_{j,j'} : 1 \leq j \leq \ell^{\mathcal{R}^{\text{id}}}\}$ and j' be the formula index for an identity relationship. Thus, the latter set comprises all specification formulae of all identity relationships that party \mathbf{A} has established. The sets of formulae Φ^{idf} and Φ^{id} specify all identifier and identity relationships of the party from a data model perspective.

\mathbf{A} selects the set $\bar{\Phi}^{\text{idf}} \subseteq \Phi^{\text{idf}}$ comprising the formulae $\mathbf{t}_i^{\text{idf}}$ for all identifier objects of identifier relationships comprising a metadata predicate $\Diamond(\mathbf{t}_i^{\text{idf}}, \text{"estWith"}, \mathbf{t}_{\text{sid}_{\mathbf{B}}})$, that is, an identifier object established with the interaction partner using the pseudonym identifier $\mathbf{t}_{\text{sid}_{\mathbf{B}}}$ in the current interaction. Furthermore, for a formula to be part of ψ' , the corresponding identifier relationship should not comprise a metadata predicate that flags it as disabled.

Similarly, \mathbf{A} constrains the set Φ^{id} to $\bar{\Phi}^{\text{id}} \subseteq \Phi^{\text{id}}$ by only considering formulae of identity relationships which are active, that is, do not have metadata associated that flag them as revoked, expired, superseded, or otherwise disabled.

Let $\phi_1^{\text{crd}}, \dots, \phi_{\ell^{\text{crd}}}^{\text{crd}}$ be the object specification formulae corresponding to the conditionally released identities $\mathbf{t}_1^{\text{crd}}, \dots, \mathbf{t}_{\ell^{\text{crd}}}^{\text{crd}}$ of the formula that are available in metadata predicates associated with formulae of data track entries at \mathbf{A} and the set $\hat{\Phi}^{\text{crd}}$ the set of those formulae. Let, analogously, be $\hat{\Phi}^{\text{oid}}$ be the set $\{\phi_1^{\text{oid}}, \dots, \phi_{\ell^{\text{oid}}}^{\text{oid}}\}$ of object specification formulae corresponding to available opaque identities $\mathbf{t}_1^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$. Those formulae can be easily obtained from the data track entries for previously released formulae for the cryptographic protocols through which the cryptographic objects have been computed through looking them up using the term names in the request formula.

For to-be-generated conditionally released identities $\mathbf{t}_{\ell^{\text{crd}}+1}^{\text{crd}}, \dots, \mathbf{t}_{\ell^{\text{crd}}}^{\text{crd}}$ and opaque identities $\mathbf{t}_{\ell^{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ it can be assumed that such identities with arbitrary attributes and being of arbitrary type as induced through their constituting attribute tuples can be created as it is required for fulfilling a request ψ , unless the formula makes a contradictory statement about their attributes, and thus they can be considered available in the computation of finding solutions to the request ψ . The terms used to refer to those objects are random terms and are replaced with terms based on the cryptographic material in $\phi_{\mathbf{B}}$ once the required cryptographic protocol steps have been executed by party \mathbf{A} .

Let \mathcal{O} be a theory modeling the composed ontology over which reasoning by the party is done in our logic. This comprises the complete ontology the party intends to consider for the policy matching. The ontology is composed from ontologies of the different ontology types as discussed in Sec. 4.10. How those ontologies are provided by ontology providers and obtained by the party is also discussed in the referred-to section. Using the ontology \mathcal{O} allows the party to leverage ontology-defined knowledge for its automated deductions.

We henceforth denote the set of formulae of the part \mathcal{O}^{tPy} of the composed ontology \mathcal{O} for specifying the identifier objects of certifiers and conditional recipients as $\Phi^{\text{idf/ont}}$ and the set of identities of certifiers and conditional recipients as $\Phi^{\text{id/ont}}$. This includes the certifiers for the dummy identities being specified. Furthermore, we denote the set of specification formulae for dummy identities from the part \mathcal{O}^{dy} of the ontology as Φ^{dy} .

Further sets of specification formulae, in addition to those in \mathcal{O} resulting from the composition with \mathcal{O}^{tPy} , make assertions about the certifier identifier objects and identities that the party holds locally in its profile data—related to the certifiers of its identity relationships and possibly further certificates. As well, corresponding formulae make statements about conditional data recipients. We denote the sets for the formulae specifying identifier objects and identities of those third parties as $\Phi^{\text{idf/cert}}$ and $\Phi^{\text{id/cert}}$, respectively. Those formulae may be additionally resembled through formulae in the ontology, without any effect. New formulae are added to those sets whenever a new certifier is encountered for establishing an identity relationship, or new data recipient is encountered in a data release protocol. Holding those formulae outside of third-party-issued ontologies is required in order to support that certifiers not captured in a third-party-issued ontology be considered in the automated deduction process. This is a prerequisite for openness and scalability of the identity system as it removes strong assumptions on the reliance on third parties for the logic-based reasoning to work. Additionally, it can be seen as crucial for the bootstrapping of such open system until the overall required ecosystem has emerged.

Also, the party can use specification formulae $\widehat{\Phi}^{\text{dy}}$ for dummy identities in addition to what the ontology \mathcal{O} specifies, again facilitating openness. Those should be aligned with the specification formulae of the party specifying certifiers.

5.4.6.2 Output

The output of the matching algorithm is a set Ω of solutions ω'_i . Each solution comprises a formula ψ'' , the identifier relationship specification formulae $\widehat{\Phi}^{\text{idf}} \subseteq \Phi^{\text{idf}}$ and identity relationship specification formulae $\widehat{\Phi}^{\text{id}} \subseteq \Phi^{\text{id}}$ required for fulfilling the request formula ψ , specification formulae $\widetilde{\Phi}^{\text{dy}} \subseteq \Phi^{\text{dy}} \cup \widehat{\Phi}^{\text{dy}}$ for the dummy identities, zero-knowledge proofs related to which are to be simulated, and the generated private entities $\chi_{\ell^{\text{oid}}+1}^{\text{dfc}}, \dots, \chi_{\ell^{\text{oid}}}^{\text{dfc}}$ and $\chi_{\ell^{\text{crid}}+1}^{\text{enc}}, \dots, \chi_{\ell^{\text{crid}}}^{\text{enc}}$ for the to-be-generated opaque and conditionally released identities $\mathbf{t}_{\ell^{\text{oid}}+1}^{\text{oid}}, \dots, \mathbf{t}_{\ell^{\text{oid}}}^{\text{oid}}$ and $\mathbf{t}_{\ell^{\text{crid}}+1}^{\text{crid}}, \dots, \mathbf{t}_{\ell^{\text{crid}}}^{\text{crid}}$, respectively.

The matching algorithm computes for each such $\mathbf{t}_i^{\text{oid}}$ the set of tuples specifying

its attributes based on the request formula ψ as well as the term $\mathbf{t}_i^{\text{oid}}$ for referring to it. The tuples are stored as part of χ^{dfc} , while the cryptographic data is not contained there and added in the cryptographic protocol. Analogously, the algorithm computes the set of attribute tuples and term $\mathbf{t}_i^{\text{crid}}$ for each of the to-be-generated conditionally released identities $\mathbf{t}_{\ell^{\text{crid}}+1}^{\text{crid}}, \dots, \mathbf{t}_{\ell^{\text{crid}}}^{\text{crid}}$. The tuples are stored as part of χ_i^{enc} for each $\mathbf{t}_i^{\text{crid}}$ and the cryptographic data is generated during the cryptographic protocol only.

Each χ_i^{dfc} comprises the attribute tuples and structure for $\mathbf{t}_i^{\text{oid}}$ and corresponds to an object specification formula ϕ_i^{oid} that specifies the attribute values of $\mathbf{t}_i^{\text{oid}}$ through a sequence of predicates connected through the \wedge -connective. Let the set of formulae Φ^{oid} be defined as $\Phi^{\text{oid}} := \{\phi_i^{\text{oid}} : \hat{\ell}^{\text{oid}} + 1 \leq i \leq \ell^{\text{oid}}\}$. Analogously, each χ_i^{enc} comprises the attribute tuples and structure for $\mathbf{t}_i^{\text{crid}}$ and corresponds to a formula ϕ_i^{crid} that specifies the attribute values of $\mathbf{t}_i^{\text{crid}}$ through a sequence of predicates. The formula set Φ^{crid} is defined as $\Phi^{\text{crid}} := \{\phi_i^{\text{crid}} : \hat{\ell}^{\text{crid}} + 1 \leq i \leq \ell^{\text{crid}}\}$. The formulae in Φ^{oid} and Φ^{crid} are object specification formulae, analogous to the formulae of an identity relationship specifying the corresponding identity.

A *solution formula* ψ'' is the target formula of the computation and refers to identifier objects and identities, specification formulae of which are part of the output, and similarly for dummy identities. The constant terms used in the formula are those referred to in the specification formulae. Also, it refers to all newly generated opaque and conditionally released identities using the terms indicated in the above discussions. A result formula does not comprise (free) variables and thus makes a concrete identity statement and can be proven with our cryptographic credential protocols.

The formula ψ'' comprises $\langle \rangle$ annotations around predicates of the formula specifying for which predicates the party needs to hold cryptographic material to create a cryptographic proof for the truth of the predicate. For the other predicates, a zero-knowledge proof simulation of the cryptographic proof will be generated, this being indistinguishable from a proof to the verifier of the protocol.

The relation between a result formula ψ'' and the inputs to and parts of the output of the matching process is formally defined, in terms of the underlying logic calculus, through the sequent (5.9), where $\mathcal{O}^* = \mathcal{O}^{\text{ax}} \cup \Phi^{\text{idf/ont}} \cup \Phi^{\text{id/ont}}$.

$$\widehat{\Phi}^{\text{idf}}, \widehat{\Phi}^{\text{id}}, \widetilde{\Phi}^{\text{dy}}, \widehat{\Phi}^{\text{crid}}, \widehat{\Phi}^{\text{oid}}, \Phi^{\text{crid}}, \Phi^{\text{oid}}, \Phi^{\text{idf/cert}}, \Phi^{\text{id/cert}} \vdash_{\mathcal{O}^*} \psi'' \quad (5.9)$$

That is, the sequent comprises all prerequisite formulae from which to syntactically derive ψ'' . Of the ontology \mathcal{O} , only the system-defined axioms \mathcal{O}^{ax} are required because of the rule for deducing the type predicate from a type attribute, as well as the sets $\Phi^{\text{idf/ont}}$ and $\Phi^{\text{id/ont}}$, that is, the specification formulae for identifier objects and identities of third parties, which are not comprised in $\Phi^{\text{idf/cert}}$ and $\Phi^{\text{id/cert}}$, e.g., because of the party not having an identity relationship with a party specified therein and thus having no profile data entry for the third party, which is a common case for dummy identities to be used in branches of a disjunction. Other parts of the ontology are not required for the deduction of the data statement ψ'' from the input formulae. The intuition behind this is that the formulae on the left side, except for $\widetilde{\Phi}^{\text{dy}}$, specify the portfolio and non-portfolio parts of the repository needed for

computing a cryptographic proof and items generated during the matching process of a data request to find a result—the set $\tilde{\Phi}^{\text{dy}}$ specifies the dummy identities. Not using any further parts of \mathcal{O} as outlined implies that the relation between the formulae on the left side of the sequent and ψ'' does not require any of the ontology-based abstractions, except for the typing. Expressed differently, ψ'' is expressed without any abstractions, which is required for a formula to be provable with credential protocols.

The formula ψ'' comprises $\langle \rangle$ annotations around predicates⁵⁴ that are to be proven using cryptographic material of the party. Those predicates are the predicates which refer to identifier objects, identities, conditionally released identities, and opaque identities described through formulae in the corresponding sets in the sequent as well as the predicates expressing the type and certifier of referred-to dummy identities in the top-level \wedge -node of the formula, and required certifier identifier objects and identities.

For ensuring that \mathbf{A} can generate a cryptographic proof for ψ'' , the $\langle \rangle$ -fulfillment condition of Def. 5.1 needs to hold for ψ'' .

Definition 5.1 ($\langle \rangle$ -fulfillment) *A formula ψ'' is $\langle \rangle$ -fulfilled in \mathfrak{L} if and only if, when constructing a formula $\hat{\psi}''$ from ψ'' by replacing every $\langle \rangle$ -annotated predicate of ψ'' with the constant **true** and every other predicate with the constant **false**, the relation $\hat{\psi}'' \vdash \text{true}$ holds in \mathfrak{L} .*

Note that this definition allows that more than the minimum necessary predicates for fulfilling the definition be $\langle \rangle$ -annotated in ψ'' , that is, that the formula is “over-fulfilled” with respect to cryptographic material underlying the referenced objects.

Let Θ be a sequent of the form of (5.9) with the assumptions as discussed for the sequent and a formula ψ'' of a fragment of $\mathcal{L}^{\text{p/g}}$ as its conclusion. Then, the $\langle \rangle$ -fulfillment property of Def. 5.1 holds for the formula ψ'' . This follows from the semantics of our logic, the definition of the sequent being based, among other things, on the portfolio of the party and parts from the ontology, and the constraints imposed on $\mathcal{L}^{\text{p/g}}$ and thus any fragment thereof.

We are interested only in the interpretation \mathcal{I}^δ corresponding to a system in a given system state induced by the so-far executed protocols as explained in Sec. 4.11 on the language semantics because it relates to the cryptographic material a party holds to prove a formula correct using protocols. Expressed differently, this means that all $\langle \rangle$ -annotated predicates need to be provable with underlying cryptographic material the party holds, that is, its identifier and identity relationships, as well as publicly available material. That is, a $\langle \rangle$ -fulfilled sequent in this setting can always be proven to be true using cryptographic protocols by the party.

Definition 5.2 ($\langle \rangle$ -minimal) *A $\langle \rangle$ -fulfilled formula ψ'' is $\langle \rangle$ -minimal, if and only if no $\langle \rangle$ annotation of a predicate can be removed with the formula remaining $\langle \rangle$ -fulfilled.*

⁵⁴Allowing also for sub-formulae to be $\langle \rangle$ -annotated may be required in a protocol setting where an identity relationship can comprise statements with disjunctions. We see this currently only being efficiently possible with protocols with online certifiers realizing such identity relationships.

Intuitively, this definition means that no predicates not required for constructing a cryptographic proof for ψ'' are $\langle \rangle$ -annotated. Formulae not fulfilling this property can result, for example, when the party's portfolio can fulfill a branch of a disjunction and thereby already making the formula fulfilled, while, in addition, (parts of) other branches being fulfilled by the portfolio.

Definition 5.3 (Portfolio minimal) *A sequent of the form (5.9) with a $\langle \rangle$ -annotated formula ψ'' is portfolio minimal if and only if the sequent holds and no formula in the sets of formulae $\widehat{\Phi}^{\text{idf}}$ and $\widehat{\Phi}^{\text{idv}}$ can be removed with the sequent (5.9) still being fulfilled.*

Definition 5.3 means exactly what it defines, namely that no superfluous object specification formulae are contained in the set $\widehat{\Phi}^{\text{idf}}$ comprising the identifier specification formulae and the set $\widehat{\Phi}^{\text{idv}}$ comprising the identity specification formulae. When providing the metadata input to the RelData protocol, the cryptographic objects related to identifier objects and identities are chosen to correspond to a solution with a portfolio-minimal sequent.

When generating a cryptographic protocol through our run-time protocol generation approach of Chapter 6, a $\langle \rangle$ -minimal formula ψ'' is used as statement to create the protocol for, and the formulae $\widehat{\Phi}^{\text{idf}}$ and $\widehat{\Phi}^{\text{idv}}$ of the corresponding portfolio-minimal sequent express precisely which identifier objects and identities, and thus the corresponding relationships, of the party the proof will be based on.

Let an environment \mathcal{E} be an instantiation of free variables, that is, a mapping from free variables to constant terms of the corresponding type. We relate the result formula ψ'' to the request ψ through the theory $\mathfrak{T}^{\mathcal{O}}$ induced by the ontology \mathcal{O} over the deductive system \mathfrak{L} and environment \mathcal{E} as shown in (5.10).

$$\psi'' \vdash_{\mathfrak{T}^{\mathcal{O}}}^{\mathcal{E}} \psi \quad (5.10)$$

Note that there does not necessarily exist a bijection between the free variables referring to identities and identifier objects in ψ and constant terms in ψ'' . Due to the expansion of ontology concepts in ψ , e.g., into disjunctions of sub-statements, a single free variable in ψ may be expanded to multiple constant terms referring to concrete objects in ψ'' . An ontology-defined concept that expands to a single identity in the result leads to the special case of a one-to-one mapping to a constant in ψ'' for such free variable of the request.

The sequents (5.9) and (5.10) for a formula ψ'' formalize how the input to the matching algorithm and the ontology \mathcal{O} are used for fulfilling a request ψ .

Considering the expected number of identity and identifier relationships in $\bar{\Phi}^{\text{idf}}$ and $\bar{\Phi}^{\text{idv}}$ for typical users in a practical system and the complexity of practical data requests ψ of service providers, the number $|\Omega|$ of solutions for a request can be expected to be reasonably small. One can represent the solutions by avoiding redundancies of re-representing the equal parts of different solutions and thereby preserving the structure among the elements of the solution set which can also be advantageous for representing the solutions to a human user for soliciting required input.

5.4.6.3 Towards an Algorithm

The preceding discussions on matching a party's portfolio and other repository formulae with a request have tackled the problem from a perspective purely based on the logic formalism underlying our languages. In Sec. 4.9 we have discussed options of implementing the deductive system for our logic, particularly with an account to the differences of our logic to standard first-order logic. Below, we complement those discussions with thoughts related particularly to finding satisfying formulae for data requests, thereby extending the above discussions towards an operationalization through an algorithm.

A matching algorithm \mathfrak{M} needs to find the result set Ω , or a subset thereof, given a request ψ and further inputs as discussed above. Each result of Ω has as its main components a formula ψ_i'' and environment \mathcal{E}_i' as explained in detail. We next elaborate on crucial aspects relevant for finding such formulae and corresponding environments for our logic-based approach.

Ontology-based expansions. A complication for the algorithm \mathfrak{M} is the expansion of ontology concepts to disjunctions. Take as an example an identity the issuer of which is specified to be an OECD government issuer through an ontology concept in the third-party specification sub-formula and over which predicates are expressed in the assertion sub-formula of ψ . For better data minimization, this ontology concept may be “expanded” such that it results in a disjunction of the assertion statements, each branch referring to one of the possible identities, without comprising an ontology abstraction. Such new request is intended to improve the privacy of the party by revealing even less data in a response to it by hiding the concrete identity type it uses to fulfill a transformed attribute request predicate. The case of not performing an expansion of an ontology concept results in a result comprising a fulfilling statement to one of the disjunctive branches induced by the abstraction, thereby being a valid response to the request.

Let $\iota(\psi, \mathcal{O}, par, \bar{\Phi}^{\text{idf}}, \bar{\Phi}^{\text{idv}})$ be a function taking a request ψ , ontology \mathcal{O} , parameters par , and formulae $\bar{\Phi}^{\text{idf}}, \bar{\Phi}^{\text{idv}}$ representing the applicable parts of the party's portfolio as inputs and, governed by par , expanding ontology-based abstractions in ψ . Concretely, it expands an ontology-based specification sub-formula for an identity to multiple identities satisfying such sub-formula and transforming ψ to a logically equivalent, under the theory $\mathfrak{T}^{\mathcal{O}}$ induced by \mathcal{O} , formula which expresses the relation predicates over the abstractly specified identity with disjunction semantics over the different identities fulfilling the abstraction. This defines a new formula that states the same predicates, in a disjunction branch for each of the identities fulfilling the abstraction. The function also outputs information for computing the environment \mathcal{E}_i .

Note that our formal logic does not mandate whether such ontology-based abstraction in a request be responded to through predicates over a single fulfilling identity or a disjunction over all or a subset of all fulfilling identities of the abstraction. Thus, the need for the above-discussed function ι and its parameter par arises.

The statement language supports different ways of realizing such expansions, of which we explain two next. (1) Each relation predicate or sequence of relation predicates expressed on the abstractly specified identity as part of the assertion sub-formula may be expanded to a disjunction of such predicates or predicate sequences, each expressed on one of the fulfilling identities. This approach is applicable to cases where the identity is referred to in an assertion sub-formula comprising disjunctions itself by creating a disjunction over sub-formulae, each referring to one concrete identity. The transformed formulae blow up substantially when multiple ontology-based generalizations are used. (2) For each ontology-based expansion, an opaque identity is introduced as a proxy for the different fulfilling identities for the abstractly specified identity. A disjunction is specified over the opaque identity, in each branch the referred-to attributes of one of the fulfilling identities being specified to be equal to the ones of the opaque identity. Each reference to an attribute of the identity in a relation predicate in the assertion sub-formula is replaced with a reference to the corresponding attribute of the opaque identity.

Particularly when attribute values are requested, non-trivial sub-formulae are expressed over the to-be-expanded identities, or multiple expansions are computed, approach (2) is preferable for resulting in simpler result formulae. Both approaches lead to formulae logically equivalent to the original request under the used ontology, though, being potentially very different structurally.

Registration binding predicates on the abstractly specified identity further complicate the processing as the registration binding identity has to be considered in addition. We do not present the syntactic details of the explained transformations in this work. Those expansions are a highly advanced approach to further strengthen data minimization and are supported by the logic.

Technically, finding fulfilling identities for an ontology-based abstraction can be realized by using the automated deduction capabilities of our logic for enumerating the concrete identities of the party that match a given specification sub-formula of a request, as part of the overall matching computation. The parameters *par* thereby govern how many disjunctive branches be used in the expanded formula and over which eligible identities. The formulae related to the portfolio of the party are required in order to also include the identity the party needs for fulfilling the disjunction to be created.

The result after applying such expansions to ψ following *par* is an intermediate formula referred to as ψ' . Note that a, possibly empty, subset of all the expansions are computed based on *par* and as allowed by metadata governing permissible expansions associated with the request. Note furthermore that *par* models also potential user input to the process or user preferences.

The output related to an environment \mathcal{E}_i for a ψ'_i can, for example, be based on a random choice of any suitable object in ψ' to be mapped to an object in ψ in order to not reveal the $\langle \rangle$ annotation of the formula.

Generally, it is possible that different parameter sets par_i be used for obtaining different expanded requests ψ'_i on which the further processing is performed. Also, this computation can, for obtaining the widest coverage of potential responses to the request and considering available identity relationships at the same time, be

interleaved with the remaining matching computations. Though, we assume for our explanations—for sufficing with a simpler formalism in our logic—that a single parameter set be used and a single formula ψ' be computed before the remaining matching steps. This covers also the setting of no such expansions being done by the party and thus $\psi' = \psi$. In a system, also the approach of not allowing for such expansions at all and requiring them to be made explicit already in the original request is practical.

A computed request ψ' can refer to a larger number of identities for primary identities as well as for corresponding certifier identities and certifier identifier objects through the expansion of abstract ontology concepts to disjunctive statements as outlined above. The formula ψ' can serve as the basis for establishing the formal relation in the underlying calculus \mathfrak{L} between a result formula ψ'' and environment \mathcal{E}' obtained through matching and the request ψ . We relate the result formula ψ'' to the request ψ' using the environment \mathcal{E}' as presented in (5.11):

$$\psi'' \vdash_{\mathcal{O}}^{\mathcal{E}'} \psi'. \quad (5.11)$$

Such sequent can be expressed with a one-to-one correspondence between variables used on the right side and constant terms on the left side of the \vdash -symbol which would, as outlined further above, not necessarily be the case for ψ on the right side.

Thus, a result gives rise to a formula ψ' only comprising terms that can be derived from the formulae corresponding to the cryptographic pseudonyms, credentials, other objects and relevant specification formulae \mathbf{A} holds, and dummy identities.

Deduction-based computation of ψ'' . The processing for computing a result can be formalized through an extension of the sequent notation. The left side of the sequent comprises most of the elements of the left side of the sequent (5.9), and the right side of the sequent is equal to the right side of (5.11). In addition to the discussions related to reasoning in Sec. 4.9, we require that the formula ψ'' be output as well by the automated deduction procedure as presented in (5.12).

$$\widehat{\Phi}^{\text{idf}}, \widehat{\Phi}^{\text{id}}, \widehat{\Phi}^{\text{dy}}, \widehat{\Phi}^{\text{crd}}, \widehat{\Phi}^{\text{oid}}, \Phi^{\text{idf/cert}}, \Phi^{\text{id/cert}} \vdash_{\mathcal{O}}^{[\psi'', \mathcal{E}']} \psi' \quad (5.12)$$

The extended sequent notation $\vdash_{\mathcal{O}}^{[\psi'', \mathcal{E}]}$ specifies that the formula ψ'' and the environment \mathcal{E}' be computed as part of the deduction procedure. For such ψ'' and \mathcal{E}' , both (5.9) and (5.11) hold. Note that the notation does not make the computation of the formula sets Φ^{crd} and Φ^{oid} explicit. Also note that the full ontology \mathcal{O} needs to be used, e.g., for resolving abstractions.

The requirement that formula ψ'' and environment \mathcal{E}' be provided as output, including its $\langle \rangle$ annotations, necessitates an extension of standard deduction mechanisms. So does the output of the subset of the input formulae based on which ψ'' can be derived under the used theory. The deduction mechanism furthermore needs to ensure that the formula ψ'' be generated following the syntactical constraints of the relevant fragment of $\mathcal{L}^{\mathbf{P}}$. Backtracking techniques can be employed for sampling the whole solution space for a given set of input parameters.

Using \mathcal{E}' and the information output by the algorithm ι for expanding ontology-based abstractions, the environment \mathcal{E} can be computed. The sequent (5.10) holds with respect to this environment \mathcal{E} . As already noted earlier, this environment may require, depending on expansions performed in the computation of ψ' , that it be computed using an approach that creates a randomized mapping.

Technically, ontology-based expansions could be part of the above deduction process. Through standard disjunction introduction rules, the desired formulae could eventually be derived through reasoning, though, the search space is huge and required extensions to a reasoning engine substantial. Our opinion is that having the expansion separated keeps the reasoning more closely aligned to standard logic and the overall system substantially simpler.

The beauty of the proposed solution of using an extension of the deductive system \mathfrak{L} for our logic for finding fulfilling formulae for a given data request shows in the complexities of the matching process being handled through the deductive system. As an example, this comprises the proof of logic equivalence of disjunctions obtained by applying ι to the corresponding ontology-based concepts. This shows the tight integration of the overall data modeling and system with the underlying logic.

5.4.6.4 Discussion

Intuitively, a formula ψ'' from a fragment of $\mathcal{L}^{\mathbf{P}}$ can comprise statements about multiple pseudonyms and statements referring to attributes of multiple credentials the party holds, thereby revealing partial attribute information of attributes of multiple credentials, as well as dummy identities for which the party does not hold credentials. Predicates expressed over those can reveal partial information on the attributes, relate opaque attributes to credential attributes, and release attributes under conditions to designated parties using conditionally released identities. Certifiers and conditional data recipients are specified through assertions about further identifier objects and identities.

For a real-world system, restricted fragments of both the data request and representation languages $\mathcal{L}^{r/g}$ and $\mathcal{L}^{p/g}$ and thus the required matching functionality can be implemented, thereby resulting in a less expressive but simpler system. For first practical deployments of user-centric identity management system based on private certificate technology as the data release protocols, for example, no support of disjunctions in a data statement is a reasonable approach, while this functionality can be added at a later stage as an incremental deployment.

The data recipient also uses automated deduction in a theory induced by a composed ontology $\mathcal{O}^{\mathbf{B}}$ in the calculus \mathfrak{L} . Its computations can be accelerated if the environments \mathcal{E}' specifying the free variable assignment with respect to the request, the chosen expansion ψ' , and \mathcal{E} are provided together with the response formula ψ'' .

Sequent (5.13) is evaluated first to check whether the response ψ'' fulfills ψ' . Next, \mathbf{B} can efficiently check whether ψ' is a valid expansion of the request ψ . This together establishes validity of the sequent (5.14) and thus that the request is fulfilled.

$$\psi'' \vdash_{\mathcal{O}^{\mathbf{B}}}^{\mathcal{E}'} \psi' \quad (5.13)$$

$$\psi'' \vdash_{\mathcal{O}^{\mathbf{B}}}^{\mathcal{E}} \psi \quad (5.14)$$

When not using this accelerated approach to verification, additional functionality must be reflected in an extension of the automated deduction procedure of the recipient such that the recipient can derive a valid environment as part of a deduction as shown in sequent (5.15) and prove the sequent correct, without using any auxiliary input for accelerated computation.

$$\psi'' \vdash_{\mathcal{O}^{\mathbf{B}}}^{[\mathcal{E}^{\mathbf{B}}]} \psi \quad (5.15)$$

5.4.6.5 Open challenges

Ontologies and disjunction requests. Regarding the use of ontologies, a challenge in terms of system openness arises from the way a response relates to a request through an environment. Assuming that a party does not have dummy identity ontologies available at its disposition for identity types referred to in a request it does not have identity relationships for, problems arise for a request comprising a disjunction where the party intends to fulfill one branch of the disjunction with its portfolio and does not have formally represented or other information on identity types referred to in the other branches.

The problem consists when a disjunctive request is fulfilled with a response comprising only conjunction as connective because, through inherent properties of the data representation, the type and certifier of all referred-to identities are specified as part of the top-level conjunction in a request or response preamble. Without ontology knowledge over the identities the party does not have an identity relationship for, the sequent for relating the response and request does not hold in the deductive theory and portfolio used by the party.

The problem can be addressed in a practical system by representing a request that comprises a disjunction over k different options as k different requests communicated in a request message of which the party chooses one. Another option is to change the request syntax such that a request can have a disjunction at the top level and each disjunctive branch comprising only the conjunction connective. This would eliminate the need for having type and certifier information about identities not backed by the party's identity relationships in the top-level conjunction of the response formula.

This problem highlights some of the issues that can arise when using a fully logic-based language for representing data and the challenges resulting from using an open system and the expressivity we support with our approach.

Deduction system. When computing a fulfilling data statement for a request with the properties as explained further above, one aspect to be considered is that

the resulting formula needs to fulfill the language constraints of the language for the respective technology being used. This brings technology dependence into the computation and can lead, depending on the supported technologies, to differently constrained data statements in line with what the underlying technology can prove. Our discussions have so far assumed a single request-response language pair (\mathcal{L}^r/c , \mathcal{L}^p/c), and thus technology, being used.

Two design approaches for realizing technology-dependent matching computations are the following: (1) A single matching engine that can handle multiple technologies and their constraints, or (2) one matching engine per technology to account for the technology-specific constraints. The matching engine thereby needs to consider technology capabilities as well as aspects of collateral release of the respective technologies for finding data-minimizing results.

5.4.7 Conventions for Requests

We next discuss conventions for data requests for a given request pattern which should be followed for avoiding subtle logic-related issues. A request ψ that is to request an attribute value of a specific attribute of either of a set of eligible identities can be expressed using different approaches.

The optimal approach is to express the type and certifier of a single identity variable through a generic way using disjunctions or ontology-based generalizations in the request preamble and express the attribute request over the attribute of this identity. This captures only a subset of possible requests for requesting attribute values of different identities.

When different identities have different attribute names for the same concept, which is possible in an open system without a centrally governed management of attributes, the optimal approach is not applicable. For example, this is the case when different identity types and certifiers use different attribute names for the same conceptual attribute, e.g., *lastName* and *familyName*.

The not-recommended approach in this case is to request a disjunction where in each branch of the disjunction an attribute value of the attribute of one of the eligible identities is requested. When having multiple free variables in different branches of the disjunction directly associated with multiple identities, this will result in the following problem when matching with a party's repository: For the one occurrence of a free variable that corresponds to the identity the party has an identity relationship for, the attribute value will be revealed in a response. Though, for all identities with dummy certificates as underlying cryptographic material, the values need to be chosen appropriately, thereby introducing complications: Either, those values are chosen randomly, thus making clear immediately through which identity the value is provided and not comprising clean response semantics, or requiring that this request pattern be recognized by the system and the values be chosen to be the same as the one of the identity for which the party has a corresponding certificate, thereby concealing through which identity the attribute value has been released.

The recommended convention to follow in this case is to express the attribute request on a to-be-generated opaque identity represented through a free variable in

the top-level \wedge -node of the formula and relate the attributes of this opaque identity to the eligible identities in disjunction branches. Then, a single free variable which will be assigned the requested attribute value in a response is exposed in the request. This resolves the above issue, though leads to another subtle issue: The party needs to choose one appropriate attribute name for the requested attribute, out of potentially multiple such. This is outside of the scope of our logic and needs to be handled through the application layer or extensions.

The above-discussed issue is not a problem related to the request language or its formalization, rather the correct use of such generic logic-based language. As a general convention derived from those discussions, no attributes should be requested to be revealed in disjunction branches in the attribute assertion sub-formula. Rather, the above approach of using an opaque identity over which the request is expressed and which is related to the actual identities in the different disjunction branches should be used. This is a convention and not enforced in the language, though, automated tools could check requests for this to hold.

A more generic problem related to the use of ontology-based generalizations in an open system is the above issue that different attributes of different identities fulfilling the ontology concept may have a different attribute name, although referring to the same conceptual attribute, e.g., *lastName*. This implies that relation predicates cannot be expressed over a single attribute of this identity. Such situation is unavoidable in open systems without a central party dictating attribute semantics.

One partial solution to this problem is to standardize widely used attribute names and their meaning, considering the needs of a largest-possible set of stakeholders. This should mitigate the issue to some extent, though, not completely. A technical solution is to extend our languages to allow for a formal mapping to specify that different attributes refer to the same “conceptual attribute” and relate predicates to one of the attribute names or a new name for the conceptual attribute. Variables for attributes are possible with respect to first-order logic, thus the extension is expected to fit into our logic framework without major changes of the language.

5.4.8 Collateral Release

Let ψ be a request for data expressed in a fragment of the language $\mathcal{L}^{r/g}$. This request can be fulfilled through any of a plurality of valid responses Ω by parties in a system, where ω'_i is one response. For one result formula ψ''_i , the sequents (5.9) and (5.11) need to hold with respect to an environment \mathcal{E}'_i instantiating free variables and an expanded request ψ' . The possible responses can differ greatly in terms of data minimization, that is, whether and how much additional information is released on top of what is required to fulfill ψ .

For example, a greatly sub-optimal approach in terms of minimizing data release is an attribute value being stated in a response for which only an inequality predicate over the attribute is requested. Providing the value may reveal substantially more information than requested through an inequality predicate. Traditional certificate technology is an example for a mechanism with which certified attribute values need

to be released, instead of also allowing for predicates to be expressed over them. The best response ϕ' to a request ψ w.r.t. data minimization and thus privacy is one that reveals as little as possible—ideally no—information in addition to what is required to fulfill ψ . In a system which does not support generalizations through ontologies or the logic-based approach to subtyping in a request as discussed in this work, a request essentially specifies precisely which identities of which types certified by which certifying parties need to be used in a response. In such setting, the best response ϕ' in terms of data minimization is such that resembles exactly the predicates requested in ψ , that is, ϕ' is obtained through term replacement of free variables in the request through constants induced through an environment \mathcal{E} without the need of considering ontology concepts.

Ontology-based generalizations as supported in our data representation complicate matters because an identity, certifier, conditional recipient, or property predicate in a request can be fulfilled with various concrete instances in a valid response as governed through the ontology. Also, an ontology-based expansion ι adds additional degrees of freedom for reducing the release of information, e.g., regarding the concrete type and certifier of the identity held by the party and used in the proof through expanding predicates expressed on an ontology concept to a disjunction over multiple such over different identities. Additional complexity in deciding which of the possible responses is most data minimizing lies in the anonymity sets the party resides in in the view of the data recipient for the different identities it can use for one abstractly specified identity in a request. All this makes it substantially harder to determine in a programmatic manner which of the possible responses is the best in terms of data minimization. The best data-minimizing response statement ϕ' for ψ in such general setting we operate in is one which utilizes the possible ontology-based expansions for minimizing data release, uses credentials and attributes such that the party resides in the largest-possible anonymity set in the view of the recipient, and employs credential technology to the widest extent possible to prove a response formula comprising the predicates of the request, while not replacing request predicates with more-revealing ones, e.g., an inequality with an equality.

Note that being able to derive an optimally data-minimizing response ϕ' , the *data-minimal response*, to a request ψ requires that the request be phrased accordingly with the cryptographic mechanisms in mind that can be used for proving correct a valid response to the request. That is, the request must not express assertions not provable with the available cryptographic techniques, e.g., non-supported predicates over attributes.

We use the concept of *collateral release* for expressing additional information being released to the data recipient or third parties through releasing a response ϕ on top of what is minimally required for satisfying a data request ψ .

Definition 5.4 (Collateral release) *Collateral release is the release of information to the data recipient of a transaction or third parties through the response ϕ or the proof protocol in addition to what is released through release and prove of a minimal response ϕ' to a request ψ .*

That is, collateral release captures the excessive information not required for ful-

filling ψ , though contained in ϕ or being otherwise released through the inherent properties of the employed mechanism and not captured in ϕ .

When creating a data statement fulfilling a request, collateral release can arise from decisions in the matching process or selection of the result that release excessive information, or the use of inappropriate technology for releasing data. For example, when using a protocol based on an online certifier for releasing a data statement ϕ , the certifier learns certain information which is not captured in ϕ and neither relevant for the data recipient to make an authorization decision. As another example, consider a response ϕ revealing more attribute information than required for fulfilling ψ , which is collateral release captured in ϕ . The latter is, for example, the case when using traditional certificate protocols for the data release instead of private certificate protocols.

For computing a fulfilling response ϕ for a request ψ based on a party's identifier and identity relationships, ϕ needs to be generated such that it takes the technology of the identifier and identity relationships into account and reflects precisely the identity statement being learnt by the other party. As an extreme example, fulfilling a simple inequality predicate over the party's date of birth attribute may require one to release all attributes of a certificate when using traditional attribute certificates, which needs to be reflected in the formula ϕ . For supporting identity relationships based on multiple different technologies thus requires that this be considered in the matching process and a result ϕ be generated accordingly. Collateral release to third parties, such as an online certifier, is never captured in the statement ϕ and is therefore additional collateral release through the employed mechanism. Reducing collateral release requires an interplay of properly authored policies, support by the matching engine, and preferences and input by the party to be considered.

The term collateral release has been originally introduced by the author in the talk for presenting the work on privacy-enhancing yet accountable access control [BCS05] as specifying the additional information leaked through the specification formula for a credential protocol, where any information beyond the pure attribute information was considered such additional information. For credential systems this has been found comprising the certifier and type information of credentials as well as what third parties learn in case of using conventional protocols for ensuring attribute integrity. For the definition of conditional release in this thesis, we include the information relevant for the trust decision in the information required to be revealed which leads to our revised definition of the term collateral release that seamlessly fits our generalized model.

5.4.9 Result Selection

The final step of computing a response to a request comprises a selection of one result ω'_i from the set of valid results Ω for fulfilling the request ψ . This can comprise a choice made by the human user, an algorithm, e.g., one operating on the user's preferences and interaction history, or a combination thereof. A combination of user and machine selection is envisioned for practical systems. The formula ψ''_i , being part of ω'_i , is released to the other party together with cryptographic elements

and a proof of its correctness. *Consent* given by the human user, as required, e.g., in European legislation [Eur95], is the final step before an instance of the RelData protocol can be executed on the chosen formula ψ_i'' to be released, or could be part of the selection process. See work by the author [CSSZ06, BCPS09] for two possible realizations for such interface for human-computer interaction in a simplified setting and this thesis for a summary of the discussions. While the further of those works focuses only on the selection of a valid response with a minimum consumption of browser real estate, the latter supports a more expressive language for data statements and considers user consent more explicitly. Both approaches, though, are based on a less expressive language than the one used in this thesis.

The outcome of the selection is a single solution ω_i' , comprising a formula ψ'' and the additional result elements as discussed earlier. The formula is exactly, using different notation, the input formula $\phi_{\mathbf{A}}$ of \mathbf{A} in a RelData protocol instance and specifies the data to be released. This formula thereby particularly determines the run-time generation of the cryptographic protocols to prove the formula correct. Other result elements of ω_i' are input to the protocol, together with additional elements such as public keys and further cryptographic material, as explained in detail earlier. The result elements of ω_i' are sufficient to obtain all further required inputs for the protocol RelData.

5.5 Discussion

A substantial fraction of the discussions in this chapter has been related to terms and their renaming throughout multiple protocol instances. The presented approach follows the idea of a party \mathbf{A} using certain terms for referring to parties, including itself, and other objects, e.g., identities, locally, and renaming certain of those terms before revealing a formula referring to the terms to another party. That is, correspondences between objects are established through relating them to the same or different parties through their subject and holder attributes in a party's local data representation. For terms referring to the subject and holder of objects, the renaming is governed through the identifier objects the party has with the recipient of a formula. Identities receive a fresh name whenever holdership of them is proven to a data recipient. This renaming ensures that no undesired linkabilities are established between the issuing and prove transaction for a private certificate and multiple proof transactions of the same certificate. The renaming reflects that, instead of the original signature of the certificate, a proof of knowledge over a randomized certificate based on it is sent to the other party. The original choice of terms for a party or object is always done by deriving the term from the cryptographic object it relates to. Technically, a logic with equality allows for such renaming and referring to the same entity with different names.

Our approach to term renaming allows a party \mathbf{A} to perform automated derivations in the underlying logic calculus in a generic way using the formulae as represented locally by having objects properly related with each other. Parties are referred to with the same term throughout different formulae in the view of the

party and thereby the intended relationships between the referred-to objects are expressed. Other parties can do reasoning over multiple formulae they have received from (the same) other party as well.

Regarding the renaming of terms in order to avoid unintended linkabilities between actions of the party, the rule is that whenever a mapping is defined in the metadata of a formula of an identifier relationship, the mapping must be used to rename locally used terms in a formula before sending the formula to the other party. The application of such a mapping is required to maintain unlinkability of the transaction to previous transactions.

5.6 Realizing the Channel Calculus

The mapping of channel transformation rules to cryptographic protocols has been discussed abstractly together with selected rules in Sec. 3.5. Further above in the current chapter, we have presented the protocols and their relation to our data model and channel model. In the current section, we present further details on how rules are implemented through the cryptographic protocols and thereby present a connection between the secure channel model and cryptographic protocols for realizing the rules. The discussion builds upon the cryptographic protocols, the interfaces of which have been presented earlier in this chapter.

A particular aspect we consider is how registrations of parties in the system and for delegations relate to the generation of new private keys of parties and corresponding private key domains. We present how the aspects of applying the data model discussed in Sec. 4.8 are realized with protocols.

The channel transformation rules express what is possible in terms of transferring security symbols and endpoint annotations from prerequisite channels to target channels using standard and advanced privacy-enhanced cryptographic protocols. Thereby, there is no one-to-one correspondance between rules and protocols: Applying a single rule may require multiple cryptographic protocols to be executed between parties, with specified constraints in terms of timing, annotation statements and security symbols. For example, rule **Auth-Cred** requires, in the general case, multiple protocol instances of protocols **EstIdtfRelReg** and **EstIdtfRel** for establishing pseudonyms and **EstIdtyRel** for issuing credentials and one instance of **RelData** for releasing data based on the pseudonyms and credentials and other objects. A single instance of an executed protocol can be part of multiple instances of the application of a rule. For example, a single instance of **EstIdtyRel** for issuing a credential can be used for multiple applications of rule **Auth-Cred** for transferring authentications between channels. When a rule requires multiple cryptographic protocols to be executed, the *timing* annotations $t_i[t_j]$ of channels define when the cryptographic protocols are required to be executed, and thus also their dependencies.

Note that, when using cryptographic protocols that only support the unlinkability property of issuing and proving holdership of a credential for a single show, that is, *single-show unlinkability*, a different mapping of the rules **Auth-Cred** and **AuthDele-Cred** is required, namely requiring a credential re-issuing for every use of a

credential, also for delegation credentials. A prominent example for such protocols is the *UProve* scheme building on the credential system of Brands [Bra00] based on blinding signatures in the issuing transaction. Such systems do not support the multi-show unlinkability property of the cryptographic protocols we build on in this thesis, which break the linkability to the issuing during the show protocol of a credential.

Regarding the notation in this section, we use generic notation that makes the parties involved in created objects explicit in the notation, e.g., in order to more easily see correspondance of objects to the same registration domains. For this reason, notation differs to the notation used in the chapter on the secure channel model.

5.6.1 Cryptographic Objects

We first present the cryptographic realizations of the identity management concepts of Sec. 4.8 using a pseudonym and credential system. The different kinds of pseudonyms and credentials realize the concepts introduced in Sec. 4.8.

5.6.1.1 Pseudonyms

A cryptographic pseudonym is computed as a cryptographic commitment of a tuple ν of cryptographic values. Without loss of generality, we assume that the first element $\pi^1(\nu)$ is always a private key of the party holding the pseudonym. Further elements depend on the kind of pseudonym being considered.

A *registration pseudonym* $\mathsf{p}_{A,C_i,\ell_r}$ of a party **A** with a registration certifier \mathbf{C}_i for the party itself as the subject of the pseudonym is computed on the tuple $\nu_{A,C_i,\ell_r} = (x_{A,C_i,\ell_r})$ using the Pedersen scheme [Ped91], with the value x_{A,C_i,ℓ_r} being a private key of the party generated as part of this protocol. This results in an information-theoretically binding and computationally hiding commitment of ν . The pseudonym corresponds to a regular registration pseudonym as defined by Camenisch and Lysyanskaya [CL01].

A *delegation registration pseudonym* $\mathsf{p}_{A,D,C_k^*,\ell_d}$ of party **A** with delegation certifier \mathbf{C}_k^* is computed over the tuple $\nu_{A,D,C_k^*,\ell_d} = (x_{A,C_i,\ell_r}, z_{A,C_k^*,\ell_d})$, where the first value is a private key of a registration pseudonym and z_{A,C_k^*,ℓ_d} is a private key of **A** generated for the delegation. $\mathsf{p}_{A,D,C_k^*,\ell_d}$ is a pseudonym under which party **A** is known to the delegation certifier which can issue one or more delegation credentials on $\mathsf{p}_{A,D,C_k^*,\ell_d}$ to **A** on behalf of a delegater **D**. The key z_{A,C_k^*,ℓ_d} is denoted *delegation private key*, held by **A**, and used by **A** for all pseudonyms and credentials to be established and obtained talking about the delegater **D** in the context of this delegation registration pseudonym.

A *delegation pseudonym* $\mathsf{p}_{A,D,B_l,\ell_b}$ of a party **A** with a party **B**, with **A** as holder and a delegater **D** as subject is computed over the tuple $\nu_{A,D,B_l,\ell_b} = (x_{A,C_i,\ell_r}, z_{A,C_k^*,\ell_d}, y_{A,B,\ell_b})$, where the first element is **A**'s private key to use, z_{A,C_k^*,ℓ_d} the delegation private of **A** to use, and y_{A,B,ℓ_b} a random value for obtaining the information-theoretic hiding property of the private keys.

A *non-registration pseudonym*, or *derived pseudonym*, $\mathbf{p}_{A,C'_j,\ell_o}$ of party **A** with party **C'** for the party as subject is computed as commitment over the tuple $\nu_{A,C'_j,\ell_o} = (x_{A,C_i,\ell_r}, y_{A,C'_j,\ell_o})$, where x_{A,C_i,ℓ_r} is the private key corresponding to a registration pseudonym $\mathbf{p}_{A,C_i,\ell_r}$ and y_{A,C'_j,ℓ_o} is a randomizing value for obtaining the information-theoretic hiding property.

Recall, that a registration or delegation registration pseudonym is established with an instance of the protocol **EstIdtfRelReg**, while a non-registration pseudonym or delegation pseudonym is established with an instance of the protocol **EstIdtfRel**. The indices ℓ_r, ℓ_d, ℓ_b , and ℓ_o range from 1 to the number of pseudonyms of the respective kind of the party **A** with the respective other party. All pseudonyms are based on the Pederson commitment scheme for single group elements or the variants thereof generalized to commitments of tuples of group elements.

5.6.1.2 Credentials

A credential is a message tuple μ and a signature over it using one of the schemes of Camenisch and Lysyanskaya [CL02b, CL04] or another scheme with comparable properties. We assume, without loss of generality, that the private key of a party is always encoded through the first attribute of a credential $\pi^1(\mu)$.

A credential $\mathbf{c}_{A,C_i,\ell_r,\hat{\ell}_r}$ issued by a registration certifier **C_i** to party **A** on a registration pseudonym $\mathbf{p}_{A,C_i,\ell_r}$, with **A** as both holder and subject, is a signature over a message tuple $\mu_{A,C_i,\ell_r,\hat{\ell}_r} = (x_{A,C_i,\ell_r}) || \chi_{A,C_i,\ell_r,\hat{\ell}_r}$, where (x_{A,C_i,ℓ_r}) is the 1-element tuple comprising the private key for creating the registration pseudonym $\mathbf{p}_{A,C_i,\ell_r}$, and $\chi_{A,C_i,\ell_r,\hat{\ell}_r}$ is a tuple comprising the remaining attributes of the certificate resulting from credential features and user attributes as derived from the high-level issuing specification. Such credential is referred to as *registration credential*, or *credential* if the registration property is not relevant in the context being considered.

A credential $\mathbf{c}_{A,C'_j,\ell_o,\hat{\ell}_o}$ issued by a certifier **C'_j** to party **A** on a non-registration, or derived, pseudonym $\mathbf{p}_{A,C'_j,\ell_o}$, with **A** as both holder and subject, is a signature over a message tuple

$$\mu_{A,C'_j,\ell_o,\hat{\ell}_o} = (x_{A,C_i,\ell_r}, y_{A,C'_j,\ell_o}) || \chi_{A,C'_j,\ell_o,\hat{\ell}_o}, \quad (5.16)$$

where $(x_{A,C_i,\ell_r}, y_{A,C'_j,\ell_o})$ is the tuple comprising the private key and randomizing value used for creating the pseudonym $\mathbf{p}_{A,C'_j,\ell_o}$, and $\chi_{A,C'_j,\ell_o,\hat{\ell}_o}$ stands for the remaining attributes of the certificate, analogous to above. Such credential is referred to as *non-registration credential*, or *credential* if the non-registration property is clear from the context.

A credential $\mathbf{c}_{A,D,C_k^*,\ell_d,\hat{\ell}_d}$ issued by delegation certifier **C_k^{*}** to **A** on a delegation registration pseudonym, with party **A** as holder and **D**, the delegater, as subject, is a signature over the message tuple

$$\mu_{A,D,C_k^*,\ell_d,\hat{\ell}_d} = (x_{A,C_i,\ell_r}, z_{A,C_k^*,\ell_d}) || \chi_{A,D,C_k^*,\ell_d,\hat{\ell}_d}, \quad (5.17)$$

where $(x_{A,C_i,\ell_r}, z_{A,C_k^*,\ell_d})$ is the message tuple of the delegation pseudonym and $\chi_{A,D,C_k^*,\ell_d,\hat{\ell}_d}$ are the remaining attributes analogous to the above. Such credential is referred to as *delegation credential*, or *credential* if the delegation property is not of interest in a particular context.

The indices $\hat{\ell}_r, \hat{\ell}_o$, and $\hat{\ell}_d$ range from 1 to the number of credentials of the respective type per pseudonym of party **A**. Recall that credentials are also referred to as *private certificates* in this work.

5.6.1.3 Private key domains

The private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$ of a private key x_{A,C_i,ℓ_r} comprises all cryptographic pseudonyms and credentials established or issued based on this key, with the key being their first attribute, including the corresponding registration pseudonym. A private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$ corresponds to the *registration domain* induced by the identifier relationship corresponding to the registration pseudonym within the establishment of which the private key has been generated. The registration domain comprises the identifier and identity relationships corresponding to the pseudonyms and credentials of $\mathcal{K}_{x_{A,C_i,\ell_r}}$.

The concept exists analogously for a delegation private key z_{A,C_k^*,ℓ_d} held by a party **A** and comprises all delegation registration pseudonyms and delegation credentials established with and issued by delegation certifier \mathbf{C}_k^* and delegation pseudonyms established with other parties based on this delegation private key. The concept is referred to as *delegation private key domain*. A delegation private key domain for a key z_{A,C_k^*,ℓ_d} is always a proper subset of a private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$ and denoted as $\mathcal{K}_{z_{A,C_k^*,\ell_d}}^D$.

For all cryptographic objects within a private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$, equality of the private key can be proven cryptographically for multiple of those objects without revealing the key, with the result of establishing the binding of the objects to the same private key x_{A,C_i,ℓ_r} and thus party. Analogously, this is true for equality of the delegation private key of the pseudonyms and credentials in a delegation private key domain $\mathcal{K}_{z_{A,C_k^*,\ell_d}}^D$, where equality of the delegation private key refers to equality of the delegater.

5.6.1.4 Domain Bridging

A party may have pseudonyms and credentials in different private key domains $\mathcal{K}_{x_{A,C_i,\ell_r}}$ and $\mathcal{K}_{x_{A,C_{i'},\ell_{r'}}}$. Making statements about objects from those different private key domains in a single formula and establishing that those objects are held by the same party requires an additional binding credential that establishes that the two private key domains belong to the same party. The cryptographic proof also requires that holdership of this binding credential be proven. We refer to this approach as *bridging of private key domains*.

Such bridging of private key domains and thus registration domains is supported throughout the framework and system, from the model and data representation language to the cryptographic protocols. For our model of authentication, the

corresponding concept is registration domain bridging. Bridging is not explicitly supported for delegation registration domains due to the limited use in practical scenarios and the inherent complexity.

A practical use case for such bridging of domains in a future identity system is the simultaneous use of an electronic driver's license or identity card and identity credential contained in the Secure Element of a mobile phone in a single data statement, while cryptographically establishing that both credentials belong to the same party through an additional binding credential. We think that such functionality is crucial for a future open identity system where people might have multiple registration domains and thus private key domains as in this example. Suitable issuers of private key domain credentials are either the registration issuers who provide this as additional services to the parties they issue certificates to or independent third-party issuers.

5.6.2 Protocols

We next present the relation between the cryptographic protocols of Sec. 5 and our secure channel model of Chapter 3. The notation we use most of the time is the notation from above with detailed indices for objects describing the parties involved, as well as key domains. Only for delegation issuing we revert to simpler notation used in the model in order for allowing more easily to establish a correspondence to the channel model.

5.6.2.1 Establishing Pseudonyms

Multiple channel transformation rules require the protocols `EstldtfRelReg` and `EstldtfRel` for establishing pseudonyms. A party **A** uses one of those protocols whenever it needs to establish a new cryptographic pseudonym with another party **B**. Those protocol instances are captured implicitly in the model as part of multiple channel composition rules and result in a party having established and authenticated the pseudonym and the party's capability of authenticating a pseudonym statement based on the established pseudonym at any later time towards the other party. Pseudonyms are expressed as part of the party annotation formulae of channels.

In the following, we are interested in cryptographic pseudonyms as discussed in Sec. 2.3 and Sec. 5.6.1.1 and their relations to private key domains as discussed. Certifiers act under public pseudonyms which can be realized, e.g., based on standard cryptography and PKI protocols and Fiat-Shamir-based PKI extensions thereof as discussed earlier in this thesis. Public pseudonyms are prominently required for parties acting under their public pseudonyms, such as most certifiers, service providers, and conditional data recipients, and captured equally in the model, though of less technical interest in terms of the protocol discussions.

In addition to being identifiers of parties, cryptographic pseudonyms based on our credential protocols serve another purpose as discussed further above in this section: They are based on a private key of their holder and are used for binding credentials and other pseudonyms to this private key and thus the party. When

releasing (parts of) the information of a private certificate to another party, this is done with the party acting under a different pseudonym with this other party, where it can be ensured that this be established based on the same private key of the party. Alternatively, when the new pseudonym is based on a different private key domain, bridging of private key domains can establish party equality for objects being part of different private key domains. For the secure channel model, using either the same key domain or bridging different domains is crucial for obtaining a security symbol at the channel endpoint decorating the annotation statement of the party on a channel derived through one of the rules for transferring authentications, that is, obtaining the authenticity property, when the annotation formula specifies holder or subject equality between objects from different domains.

The protocol **EstldtfRelReg** realizes the part of the channels with security symbol at party **A** towards a certifier C_i in rules **Auth-Cred** and **AuthDele-Cred** which relates to a pseudonym p_{A,C_i,ℓ_r} being established as part of a party registration. Concretely, the protocol realizes the establishment and implicit initial authentication of such pseudonym. Attributes are associated to such pseudonym through a credential on tuple $\mu_{A,C_i,\ell_r,\hat{\ell}_r}$ issued such that $\pi^1(\mu_{A,C_i,\ell_r,\hat{\ell}_r}) = x_{A,C_i,\ell_r}$, that is, the credential is bound to the private key of **A** created for the pseudonym p_{A,C_i,ℓ_r} .

The part of the rule **AuthDele-Cred** realized through issuing of (delegation) credentials to **A** requires a delegation registration pseudonym to be established as explained in more detail further below.

The target channels of rules **Auth-Cred** and **AuthDele-Cred** comprise the establishment of a pseudonym using **EstldtfRel** with **B** about **A**, and for **AuthDele-Cred** additionally a pseudonym about the delegater **D**, also established with **EstldtfRel**, all of which have $\pi^1(\nu) = x_{A,C_i,\ell_r}$ for a private key of **A** for the corresponding tuple ν of the pseudonym. The pseudonym about **D** has $\pi^2(\nu) = z_{A,C_k^*,\ell_d}$ as a delegation private key. The proof of knowledge of the pseudonyms towards **B** is done as part of an instance of the protocol **RelData**, as part of the proof of the overall formula ϕ' .

The protocols for establishing a pseudonym ensure that the new pseudonym is bound to a private key x_{A,C_i,ℓ_r} of **A** established in a registration. This requires that **A** prove holdership of a registration credential before establishing the pseudonym to establish validity of the private key, that is, to show that the party is registered in the system.

Assume party **A** has established a registration or non-registration pseudonym with a party **B**. Then, **A** can authenticate under this pseudonym towards party **B** using the channel transformation rule **Auth-Sig** for transferring the authentication to a later point in time. Using pseudonym protocols of [CL01] or [Sec10] for creating a Fiat-Shamir signature based on a proof of knowledge of the secrets behind a pseudonym corresponds to using a signature based on a traditional public/private key pair for authentication.

5.6.2.2 Credential Issuing: General Discussion

The issuing of credentials comes into play for the rules **Auth-Cred** and **AuthDele-Cred** for the transfer of multiple authentications and for delegation. Issuing by a certi-

fier \mathbf{C}_i to a recipient party \mathbf{A} is a combination of re-certifying authenticated attributes, statements authenticated through out-of-band means or through channel transformations, and new statements made by the certifier. Issuing of a credential can be modeled through a single authentic channel between the credential recipient and certifier, with the recipient being annotated with authenticated formula ϕ_i , comprising also a pseudonym. As explained in Chapter 3 and in greater detail in Sec. 5.3, the function $\mathcal{F}(\phi_i, \mathbf{C}_i, \mathfrak{N}_i)$ in rules **Auth-Cred** and **AuthDele-Cred** expresses the changes formula ϕ_i is subjected to for obtaining the formula which represents the credential to be issued. The channel $\mathbf{A} \xrightarrow{\phi_i \bullet \frac{t_{i2}[t_{i1}]}{\phi_{C_i, A}} \mathbf{C}_i}$ in the rules **Auth-Cred** and **AuthDele-Cred** represents the issuing of a credential from the certifier \mathbf{C}_i to \mathbf{A} .

For the credential system of this work, there are two kinds of certificate issuing: *registration issuing* and *non-registration issuing* of a credential. The channel model captures this through whether a channel with a security decoration is assumed to exist or derived from channels with a security symbol as decoration. Next, we explain the registration of a party in a system and the associated protocols related to pseudonyms and issuing of a credential, as well as the protocols in the non-registration case.

5.6.2.3 Registration Issuing

The cryptographic registration protocols for establishing a registration pseudonym and issuing a registration credential for a party apply, in terms of the model, in two cases. (1) Party \mathbf{A} performs a registration in the sense of the channel model, or (2) it performs a new cryptographic registration based on an authentication with the certifier obtained through channel transformation rules of the model. Case (1) is the one following the definition of registration as being based on an out-of-band process of proving identity attributes and entering the system and is the standard case for registration of a party. Case (2) extends this to obtain a more generic system, where the out-of-band authentication can be replaced with an in-system authentication. Case (2) does not fit the modeling of registration smoothly, and is considered as optional extension with additional incurred complexity.

During a registration in either Case (1) or Case (2), a registration pseudonym and credential are always generated, bound to the same private key. Without the registration credential, the pseudonym and private key would be of no use towards other parties than \mathbf{C} .

Case (1): Party Registration. As discussed in Sec 3.9, a registration comprises a party (user) \mathbf{A} becoming a registered party in the system. In a registration of party \mathbf{A} with certifier \mathbf{C}_i , \mathbf{A} first obtains a registration pseudonym $\mathbf{p}_{A, C_i, \ell_r}$ with \mathbf{C}_i through executing an instance of the protocol **EstIdtfRelReg** which comprises the generation of a private key x_{A, C_i, ℓ_r} for \mathbf{A} such that it is sufficiently ensured that only \mathbf{A} has access to this key.⁵⁵ The subject and holder functions of the to-be-established pseudonym are both held by \mathbf{A} .

⁵⁵Constraining access to the key to only \mathbf{A} can, for example, be achieved by generating and containing it in a tamper-resistant hardware token such as a smart card or Secure Element of

The key x_{A,C_i,ℓ_r} induces a private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$. With each registration a party performs with a certifier, a new registration pseudonym together with its associated private key is obtained. Each certifier may restrict each party to register once, e.g., in order to control exclusion of parties, as in the standard model of credential systems [LRSW00, CL01]. Our approach of allowing for multiple registrations with the same or different certifiers generalizes this standard approach.

Party **A** is authenticated to C_i under the pseudonym p_{A,C_i,ℓ_r} and additional attribute statements can be associated with this pseudonym through out-of-band means, e.g., through verification through C_i of a physical passport of **A**. Those statements can be modeled in our data representation language as being certified as per the out-of-band authentication. Both the pseudonym and attribute statement are modeled with a channel where **A** authenticates a statement towards C_i , comprising the pseudonym statement and the out-of-band authenticated attribute statements. Optionally, C_i can assign further attribute statements to **A**, resulting in further authentic channels from **A** to C_i , where **A** acts under the same pseudonym. Those statements can be modeled in the data representation language such that their certifier is party C_i . The resulting channels can be combined with the one obtained earlier to obtain a single channel resembling the authentication of **A** towards C_i under statement ϕ .

C_i issues a credential on the formula $\mathcal{F}(\phi_i, C_i, \aleph_i)$ which gets cryptographically bound to the established registration pseudonym p_{A,C_i,ℓ_r} of **A** with C_i by the credential being issued on, and thus bound to, **A**'s private key x_{A,C_i,ℓ_r} . The credential issuing is achieved through executing an instance of the protocol `EstldtyRel` parameterized as explained in Sec. 5.3. The credential issuing is modeled with a channel existing at a later time where **A** is authenticated again with ϕ_i and the channel direction being reversed. This channel is one of the prerequisite channels for rules `Auth-Cred` and `AuthDele-Cred` for transferring multiple authentications to other parties and delegation. The credential is member of the private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$.

A registration credential is required to allow for proving to other parties that the party has registered in the system and can, and typically will, comprise also attribute statements about the party. From an architecture perspective, it is advisable that registration pseudonyms, particularly the private key associated with them, and registration credentials be issued to a protected storage area under control of the recipient party only such as a tamper-resistant hardware token of the party. Depending on security requirements, the obtained cryptographic material can alternatively be stored without additional protection. A verifier in a `RelData` protocol can decide through its authorization policies and used ontologies which credentials of which (registration) certifiers it trusts for a given purpose.

Note that in the above modeling of registration issuing, we do not separate the establishment of the registration pseudonym and the statement **A** is authenticated under into different channels, although the pseudonym represents a transaction with a cryptographic protocol in our system, while the statement is authenticated through

a mobile handset. Technically, it may make sense to view the generation of the private key as separate protocol, though, we see it as part of the protocol for establishing a registration identifier for simplicity.

any suitable out-of-band means. Our interest is only the resulting authentic channel where \mathbf{A} is authenticated with a pseudonym and attribute statement towards \mathbf{C}_i based on which a credential issuing protocol takes place. Again, authorization policies in the system will be used to determine whether a certifier accepts a certain issuing authority, thereby not imposing limiting restrictions and facilitating an open identity system.

Case (2): Derived Authentication. Another case that may be important in practical identity systems based on credentials is that a party can perform a registration with a certifier while authenticating a statement to the certifier based on a channel transformation rule and not through out-of-band means. In other words, the authentication through rule application is considered as out-of-band authentication for this case to fit the model. This case is handled in terms of protocols analogous to the case above, with the difference that the initial channel is not assumed to exist, rather that it is established through out-of-band means. The initial authentic channel needs to be created through application of rule **Auth-Cred** or **AuthDele-Cred**, that is, within the authentication system. The above discussions also apply to this case of registration issuing.

The use case for this Case (2) is to allow a certifier to act as registration certifier while obtaining the identity information of the registree through channel transformation rules based on other credentials, instead of running a (more expensive) out-of-band authentication procedure.

5.6.2.4 Non-registration Issuing

Any non-registration issuing of a credential is discussed next with respect to the involved cryptographic protocols. First, we consider issuing of credentials which are not delegated ones. In such issuing, an existing private key x_{A,C_i,ℓ_r} of the party created during a registration with certifier \mathbf{C}_i is used for issuing the credential on. Let \mathbf{C}'_j be the certifier from whom \mathbf{A} intends to obtain a credential. \mathbf{A} creates a pseudonym p_{A,C'_j,ℓ_o} with \mathbf{C}'_j based on its private key x_{A,C_i,ℓ_r} .⁵⁶ \mathbf{A} establishes a channel where it authenticates a pseudonym with itself and attribute assertions towards \mathbf{C}'_j . The resulting channel is an authentic channel from \mathbf{A} with annotation formula ϕ'_j to \mathbf{C}'_j . Optionally, \mathbf{C}'_j may assign further attribute statements to \mathbf{A} resulting in additional authentic channels assumed to be available. As in the registration issuing, those channels can be combined to a single authentic channel between \mathbf{C}'_j and \mathbf{A} , that is, it carries a security symbol decorating ϕ'_j of \mathbf{A} . The latter is crucial in order that the new credential be issued to a properly authenticated user. Note that, as explained earlier, a security symbol cannot be obtained through only establishing a new pseudonym with \mathbf{C}'_j . Rather, it requires in addition either a

⁵⁶The choice of which private key (domain) \mathbf{A} uses is based on a decision at the identity management level and out of scope in this work. It determines the private key the new pseudonym and credential are based on and thus with which other pseudonyms and credentials of the party it can be used in formulae without assuming additional domain bridging certificates for establishing equality of pseudonyms based on different private keys.

credential from a registration certifier or a credential from a non-registration certifier who has properly authenticated, in the view of \mathbf{C}'_j , the party, that is, has obtained a security bullet on the party's side of the channel, to obtain the authenticity property for the statement on this channel. A channel with the same annotation ϕ'_j of \mathbf{A} and reversed channel direction is the channel used for the issuing of a credential. The issued credential is cryptographically bound to the party's private key x_{A,C_i,ℓ_r} , thus part of the private key domain $\mathcal{K}_{x_{A,C_i,\ell_r}}$, and thus cryptographically bound to the party.

Rule **AuthDele-Cred** requires multiple uses of pseudonym protocols, data release, and issuing: \mathbf{D} establishes, using **EstIdtfRel** or **EstIdtfRelReg** a non-registration or registration pseudonym $\mathcal{D}_{C_{k+1}}$ with \mathbf{C}_{k+1} on which it obtains a credential, bound to the same private key x_D , representing channel $\mathbf{D}^{\phi_D} \bullet \xleftarrow{t_6[t_5]} \phi_{C_{k+1},D} \mathbf{C}_{k+1}$ of the rule. \mathbf{D} furthermore establishes a pseudonym $\mathcal{D}_{C_{k+1}|a}$ with \mathbf{C}_{k+1} which is additionally established with \mathbf{A} , again based on private key x_D . Establishing the same pseudonym with two parties can be achieved through an extension of **EstIdtfRel**. Through the same underlying private key of \mathbf{D} , both pseudonyms can be revealed in a single formula towards \mathbf{C}_{k+1} and shown to relate to the same party through **RelData** for establishing channel $\mathbf{D}^{\phi_D} \bullet \xrightarrow{t_6^*[t_5^*]} \phi_{C_{k+1},D}^* \mathbf{C}_{k+1}$ for delegation initiation. \mathbf{A} establishes a pseudonym $\mathcal{A}_{c_{k+1}|d}$ with \mathbf{D} , that it later establishes also with \mathbf{C}_{k+1} , using **EstIdtfRel**. \mathbf{D} acts under pseudonym $\mathcal{D}_{c_{k+1}|a}$ towards \mathbf{A} , and signs the delegation authorization token \mathcal{Y} with a statement revealing this pseudonym. Thus, \mathbf{A} obtains an authorization token it can provide to \mathbf{C}_{k+1} under pseudonym $\mathcal{A}_{c_{k+1}|d}$. \mathbf{A} establishes with \mathbf{C}_{k+1} pseudonyms $\mathcal{A}_{c_{k+1}|d}$ with \mathbf{A} as subject and $\mathcal{D}_{[a]c_{k+1}}$ with \mathbf{D} as subject. The further is a non-registration pseudonym, the latter a delegation registration pseudonym. Both are based on x_A , while the latter additionally comprises the newly generated delegation private key $z_{A,D}$ of \mathbf{A} . As \mathbf{A} can prove holdership of $\mathcal{A}_{c_{k+1}|d}$ towards \mathbf{C}_{k+1} and the same pseudonym is signed by \mathbf{D} under $\mathcal{D}_{c_{k+1}|a}$ as part of the authorization token \mathcal{Y} , it can be ensured sufficiently that the delegation credential is issued to the right party as authorized by \mathbf{D} . The delegation credential is cryptographically bound to x_A , which $\mathcal{A}_{c_{k+1}|d}$ is based on, and $z_{A,D}$ of $\mathcal{D}_{[a]c_{k+1}}$. Pseudonyms of a party being part of the same private key domain helps ensuring that the party behind the pseudonyms is the same, and this be provable, and thus the proper credential of \mathbf{D} can be delegated to the proper party \mathbf{A} . \mathbf{A} can now create new pseudonyms with \mathbf{D} as subject with other parties. Such pseudonym $\mathcal{D}_{[a]}$, is computed on private key values $(x_A, z_{A,D})$, with a third random component for the hiding property.

Through the support of such delegation over the authority of making attribute statements, we extend the basic approach of credential systems of binding pseudonyms and credentials to a master private key, see Camenisch and Lysyanskaya [CL01], to the setting of delegation and possibly multiple private key domains and delegation private key domains. See Rule **AuthDele-Cred** of Chapter 3 for additional information on the rule for delegation and how it is realized.

A practical aspect of delegation is that a delegation certificate, the original corresponding certificate of which is issued on a hardware token, will, in many practical

scenarios, be issued as software-based certificate, bound to a sufficiently protected private key of the delegatee for obtaining a flexible delegation system. The requirements on the private key to bind it to can be expressed through the authorization policy of the delegation certifier that determines the permissible prerequisite formulae and proofs by the delegatee (certificate recipient) for protocol EstIdtyRel —see Sec. 5.3. All language mechanisms, e.g., ontology-based abstractions, can be employed for the policy determining permissible prerequisite formulae. This shows again the integration for the specification of our system. A concrete example for this is the delegation of (parts of) an electronic identity card private certificate bound to a hardware token, where the delegation certificate is issued as a “software certificate” bound to a hardware-protected private key of the delegatee. For this example, a suitable private key is the one created by the delegatee when getting its registration pseudonym for obtaining its own electronic identity card.

5.6.2.5 Discussion

Registration and derived issuing. The main difference of registration and derived issuing of credentials is that registration issuing builds on an original authentication relationship between recipient and certifier established through out-of-band means, while non-registration issuing is based on an authentication derived through channel transformation rules. In the further case, an original binding of the registration pseudonym and credential to its recipient is achieved, while in the latter case, the binding of the pseudonym with the certifier and the issued credential to the recipient is based on an established original binding. Either case leads to the issuing of the credential to a properly authenticated recipient—the registration issuing is used for bootstrapping a system, while the non-registration issuing is used for system operation based on at least one bootstrapped user authentication per user in the system. Technically, this difference shows in either a new private key for the recipient being created, or a key created in a previous registration issuing transaction being reused for creating the new pseudonym and issuing the new credential, both of which reside in the private key domain induced by the used key.

Note that for both kinds of issuing we abstract in the channel model of Chapter 3 that credential issuing is an interactive multi-round 2-party protocol, see, e.g., prior works [CL01, Sec10]. A secure channel for executing such multi-round protocol can be easily obtained in practical settings from the available channels stated in the rules, including those resulting from the standard PKI model, using channel transformations.

Multiple registrations. By allowing for multiple registrations of a party and governing authentications through the approach of handling trust as explained in Section 3.8, we can obtain a flexible system in terms of choice of certifiers a user can register with and that verifiers can accept, which is, in our view, a crucial prerequisite for an open identity system with a plurality of players. The approach prevents the strong assumption of single registration authorities or specific properties imposed for such. All trust decisions are left to parties relying on attributes, while allowing

for trust outsourcing.

Security symbols. Multiple private key domains of a party are not explicitly captured with the syntax of the secure channel model, though they are formally expressed through using the data representation language of Chapter 4 for its annotations. The model thus allows for easily combining different authentications of a party with registration certifiers to a single authenticated formula. Thereby, the holder and subject of identifier objects and identities are represented through terms of type `pty`, with concepts abstracting from parties' private keys. Terms related to different private key domains are different in a statement and need to be related through additional logical equality predicates, proven through proof of knowledge of a corresponding bridging credential. That is, an additional refinement comes into play in the data representation language, which has been accounted for also in its semantics, the authorization policies, and cryptographic protocols.

Regarding the security of private keys of parties, we make the assumption that the party has control over them and only the party can use them for executing protocols. Using security tokens for containing private keys and user authentication mechanisms towards those tokens allows for strongly binding a private key to a party. What will always be possible is that the party gives away its authentication secret for the token or performs, in case this is not possible, e.g., for biometric authentication, an authentication for a transaction executed by someone else. Another possibility is that an attacker obtains the token and authentication capability. This is not a perfect situation, though, properly reflects the situation of what a party is able to do with its certified identity information in remote transactions. A stronger or more restrictive binding of tokens, and thus private keys, to their holders, and thus stronger authentication properties, seems unrealistic in practice. The accountability mechanisms we support disincentivize parties to give away their tokens or illegitimately execute transactions for other parties.

Recall that a security bullet on a channel endpoint means that the endpoint annotation statement is authenticated. For statements based on cryptographic objects from a single private key domain, the assumptions we make on private key security imply that the statement has been authenticated by the party being able to use the private key for creating cryptographic proofs or has been authenticated illegitimately by another party, possibly facilitated by the party.

Though, when statements over objects from different private key domains are combined in one formula, the situation gets more involved: Two users may illegitimately combine their capabilities of using their respective private keys and credentials and compute a proof of a formula over multiple of their credentials of their respective private key domains. This would amount to be a generalized variant of credential pooling [CL01]. As long as the statements over objects of the different domains are not related to each other by equating their holders or subjects and thus establishing that they are held by or are about the same party, such party equality semantics is not implied by such formula proven jointly by dishonest users. Thus, it is perceived by its recipient as a statement authenticated by a single party and with a bullet annotation, where the annotation means the formula is properly

authenticated, in this case, potentially jointly by two parties in case of dishonest users. Establishing party equality semantics requires that knowledge of a private key domain bridging credential for the private keys of the private key domains of both parties be proven. Though, this is impossible for dishonest users under the assumption that the issuer of bridging credentials must sufficiently ensure that private key domains it bridges are held by a single party. Thus, for a formula based on bridging private key domains, a bullet annotation implies that the formula has been authenticated by a single party, having access to the used private key domains, or another party having gained illegitimate access as discussed for a single key domain.

Considering those discussions, the semantics of our channel model of a bullet decoration of an endpoint annotation formula expresses exactly what is intuitively intended, particularly also for formulae based on multiple private key domains.

5.6.2.6 Releasing Data

Data release through credential protocols by a party **A** to a party **B** is part of the rules **Auth-Cred** and **AuthDele-Cred** of the channel calculus. Relevant aspects, such as the dependence of a data statement ϕ' to be released on the formulae authenticated to certifiers or the protocol interface and its relation to the data representation, have been discussed in detail earlier in this work.

In an instance of the **RelData** protocol, party **A** authenticates a formula ϕ' towards **B**, where the formula refers to identifier and identity relationships of at least one of the party's registration domains, backed by the corresponding pseudonyms and credentials as related cryptographic material. For rule **Auth-Cred**, only registration and non-registration pseudonyms and registration and regular credentials with the party as holder and subject are the corresponding cryptographic materials of referred-to objects in the formula. For **AuthDele-Cred**, also a delegation pseudonym and delegation credential for making identity statements about the delegater may be cryptographic material corresponding to referred-to objects. We have noted before that only a single identity relationship about a delegater can be considered in the rule, while the protocols allow for more generic operation.

The formula ϕ' may furthermore refer to conditionally released and opaque identities, corresponding to ciphertexts and commitments of tuples of attributes. A conditionally released or opaque identity being referred to may be an instantiated or uninstantiated one. In the further case, it corresponds to an existing ciphertext or commitment, in the latter case, the corresponding cryptographic object must be generated as part of the protocol instance.

Establishing the target channel of the rules **Auth-Cred** and **AuthDele-Cred** requires only a single protocol instance of **RelData** to be executed, in addition to the pseudonym and credential protocols explained further above.

Chapter 6

Run-time Protocol Generation

In this chapter we summarize our approach to generating cryptographic protocols for releasing data through the protocol **RelData** as specified in formulae expressed in the fragment $\mathcal{L}^{p/c}$ of $\mathcal{L}^{p/g}$. Each instance of the **RelData** protocol between **A** and **B** with **A** authenticating a data statement $\phi_{\mathbf{A}}$, or, after term renaming, $\phi'_{\mathbf{A}}$, requires that a cryptographic protocol implementing the semantics of such data statement, that is, allowing for an authentication of the formula, be generated and executed.

Proving a data statement $\phi_{\mathbf{A}} \in \mathcal{L}^{p/c}$ based on data-minimizing cryptographic primitives requires a cryptographic protocol that is a function of the statement. Due to the large number of possible data statements, the corresponding cryptographic protocols cannot be exhaustively defined statically. For example, if a data statement comprises an inequality predicate on a non-revealed attribute of a private certificate, the inequality is proven with an inequality proof protocol over the attribute of a private certificate without releasing the attribute. Other constellations of predicates in the formula give rise to specific cryptographic constructions for realizing them.

We use the approach of *run-time generation* of cryptographic protocol instances based on the protocol specification to be authenticated. Each formula $\phi_{\mathbf{A}}$ to be authenticated serves as input to the generation of a protocol implementing the semantics of this formula and execution thereof.

Our approach of run-time protocol generation is based on compiling the input

data statement into a cryptographic protocol implementing the semantics of the statement. The protocol is an interactive two-party protocol between the prover **A** releasing the statement and verifier **B** receiving it. When the verifier can successfully verify the protocol, it can be assured that that statement made and proven by **A** is true in the interpretation \mathcal{I}^δ of a system in a given system state which is induced by the executed protocols. This establishes cryptographic assurance about the identity assertions comprised in the statement.

Such interactive two-party protocol being generated for a statement formula builds on cryptographic protocols for data minimization, so-called zero-knowledge proof protocols [GMR85, GMR89], or, more precisely, zero-knowledge proof of knowledge (ZK-PoK) protocols [BG92], over the cryptographic elements explained in Sec. 5.4. The protocol proves knowledge of Pedersen commitments [Ped91, Sch91], that is, elements of prime-order groups, or discrete logarithm representations [CS97a], that is, tuples of elements of prime-order groups, realizing the primary identifier objects as specified in the formula. The protocol also proves knowledge of SRSA-CL or BL-CL signatures [CL02b, CL04] of the statement over attribute tuples realizing the primary identities that the party holds in its identity relationships and generates zero-knowledge proof simulation transcripts of those that the party does not hold. Opaque identities are realized through Damgård–Fujisaki–Okamoto (DFO or DF) commitments [DF02], and conditionally released identities through ciphertexts of the Camenisch-Shoup verifiable encryption scheme [CS03]. The protocol also proves knowledge of the secret preimages related to the attributes committed through the Damgård–Fujisaki scheme and encrypted through the Camenisch-Shoup verifiable encryption scheme. Additional group commitments may need to be generated for realizing the to-be-instantiated primary identifier objects, and additional DFO commitments may need to be generated to realize the logical connective structure of the assertion sub-formula in a cryptographic protocol. The technique for implementing formulae comprising disjunctions is that of partial proofs of knowledge of Cramer, Damgård, and Schoenmakers [CDS94] and later improvements thereof by Cramer [Cra96] and Cramer et al. [CDM00]. Attributes of signatures, ciphertexts, and commitments are related with each other through preimage relation proofs [CS97b]. Exact inequality proofs over committed integers are done through Boudot’s protocol reducing an inequality to the proof that an integer is non-negative [Bou00].

This approach of run-time generating protocols is in stark contrast to the approach taken in today’s deployed protocols based on conventional cryptographic schemes, e.g., protocols with online certifiers or conventional certificates. In those protocols, the cryptographic “proof” for a data statement is always computed in exactly the same way—by the online certifier issuing a signature on the statement as a response to an on-line query, or the holder of the conventional certificate proving knowledge of it using the associated private key based on standard techniques. That is, for those schemes, the proof (or verification) protocol is not a function of the statement, but is the same for all statements, and the statement is an input to this protocol. This is possible due to the less powerful identity semantics of those conventional protocols in not allowing for proving data-minimized statements.

6.1 Overview

Our approach for run-time generation builds on a multi-layer processing architecture, or stack, for both prover and verifier comprising the sequentially executed transformations $\mathcal{M}_{\mathcal{L}_{LL}}$, $\mathcal{M}_{\mathcal{APL}}$, $\mathcal{M}_{\mathcal{CPL}}$, and $\mathcal{M}_{\mathcal{CPL}^*}$ to obtain an executable cryptographic program for the prover and verifier side of the protocol, of which the prover and verifier execute their respective parts. Work on this has started already in 2008 in the scope of an implementation of a cryptographic library for the Identity Mixer credential system. The prover's and verifier's parts are structurally similar, the protocol generation being determined by the formula to be proven, and differ in certain operations that are executed, e.g., the prover operates on private cryptographic elements such as keys and certificates and needs to process the $\langle \rangle$ annotations of the formula, while the verifier obtains only shared elements and does not see the annotations on whether a party of the protocol is proven or simulated. The processing stack has five layers for handling the complexity of the transformation and allowing for optimizations at the different layers to be done. Much of the complexity is incurred through support of monotonous formulae, that is, formulae allowing for disjunctions.

The specification of an instance of the run-time protocol generation is given through a formula, henceforth referred to as ϕ . Using the notation of Sec. 5.4, this corresponds to the formula ϕ'_A obtained by A through renaming of the *holder* and *subject* terms on its protocol input ϕ_A . The first transformation $\mathcal{M}_{\mathcal{L}_{LL}}$ maps ϕ to a formula ϕ_{LL} of the language \mathcal{L}_{LL} , which is a formula still expressed in a declarative way over cryptographic objects, thereby expressing the identity semantics of ϕ through cryptographic objects instead of abstract identity management concepts. The layer related to the language \mathcal{APL} breaks with the declarative paradigm and expresses specifications of cryptographic protocols, instead of declaration of statements that hold, over the cryptographic objects. The identity-management-based specification of \mathcal{L}_{LL} is mapped, through $\mathcal{M}_{\mathcal{APL}}$, to an abstract protocol specification $\phi_{APL} \in \mathcal{APL}$ that specifies a cryptographic protocol through predicates representing abstract building blocks. The \mathcal{APL} -based protocol specification is compiled through $\mathcal{M}_{\mathcal{CPL}}$ to a detailed protocol specification ϕ_{CPL} expressed in the cryptographic protocol language \mathcal{CPL} . Through $\mathcal{M}_{\mathcal{CPL}^*}$, the \mathcal{CPL} specification ϕ_{CPL} is transformed to a specification expressed in the executable cryptographic protocol language \mathcal{CPL}^* that can be executed by a cryptographic protocol interpreter. The interpreter can realize a substantial speedup of the protocol execution by exploiting the instruction-level parallelism (ILP) inherent to the generated protocol. The cryptographic protocol layers corresponding to the languages \mathcal{L}_{LL} , \mathcal{APL} , \mathcal{CPL} , and \mathcal{CPL}^* represent protocols over cryptographic objects at different levels of abstractions. Only \mathcal{CPL}^* is a sufficiently concrete specification to be executed by a procedural protocol interpreter.

Performance is considered at multiple of the layers of the run-time protocol generation stack, e.g., through optimizing protocol representations in \mathcal{APL} or execution of protocols expressed in \mathcal{CPL}^* to exploit the advantages of computing architectures with multiple execution units.

6.2 Low-level Identity Specification Language

The language \mathcal{L}_{LL} , referred to also as *Low-level Identity Specification Language*, is a language that realizes the semantics of $\mathcal{L}^{p/c}$, through making statements over *cryptographic objects* as opposed to identity-management-level objects as $\mathcal{L}^{p/c}$ does.

Cryptographic objects and keys. A cryptographic object is an entity realizing the cryptographic concepts of commitments, signatures, and ciphertexts. A cryptographic object thereby is represented as an array of integers holding private keys or (fragments of) user and other attributes of the corresponding object referred to in $\mathcal{L}^{p/c}$. The cryptographic object also comprises cryptographic elements representing the commitment, signature, or ciphertext, and metadata such as the structure, e.g., certificate structure, for an object. The actual cryptographic elements corresponding to the cryptographic object are not of interest in \mathcal{L}_{LL} , only the corresponding integer attributes are.

For example, a private certificate $c1$ is an array, where a reference to its i^{th} integer element is denoted as $c1[i]$. The integer type int in \mathcal{L}_{LL} , \mathcal{APL} , \mathcal{CPL} , and \mathcal{CPL}^* is a (large) finite interval of integers. Predicates can be expressed over attributes of cryptographic objects and certain kinds of the objects themselves. For example, equality of an attribute $c1[10]$ with the constant “Zurich” is expressed as

$$\text{Eq}(c1[10], \text{Str2Int}(\text{“Zurich”})) , \quad (6.1)$$

where Str2Int is a function used to make the predicate more easily representable in the current presentation and encodes the length-restricted string argument to a single int constant.⁵⁷

Each identity-management-level object in the language $\mathcal{L}^{p/c}$ has a corresponding cryptographic object in \mathcal{L}_{LL} . An identifier object corresponds to the opening information of a Pedersen or group commitment and the commitment [Ped91, CS97a] in \mathcal{L}_{LL} , an identity to a private certificate, that is, the attribute tuple of a private certificate—or credential—based on the SRSA-CL [CL02b] or BL-CL [CL04] signature protocols together with the signature,⁵⁸ an opaque identity to the opening information of a Damgård–Fujisaki–Okamoto commitment and the commitment [DF02], and a conditionally released identity to the plaintext tuple of a Camenisch–Shoup ciphertext and the ciphertext [CS03]. In \mathcal{L}_{LL} , only declarative statements over the attribute tuple of a cryptographic object are of interest, while the cryptographic elements are used only in the specifications of cryptographic protocols in the following layers generated from a formula in this language.

Note that the prover knows both the attribute tuple and the cryptographic elements of a cryptographic object, whereas the verifier only receives cryptographic

⁵⁷In the general case, a string can be mapped to multiple integers, where the mapping is specified through the function \mathcal{M} of the certificate structure corresponding to the certificate type of the identity the attribute is part of.

⁵⁸Also Brands’ credentials [Bra00] can be supported as an additional primitive by the run-time generation system, though, with the important difference that they do not have the multi-show unlinkability property, for which reason they are less interesting to us.

elements related to it, metadata, and gets to know attribute information as expressed through the \mathcal{L}_{LL} formula. A cryptographic object is always referred to through the same term as used in the $\mathcal{L}^{P/C}$ formula for referring to the high-level object it corresponds to. Cryptographic keys are expressed in a formula in \mathcal{L}_{LL} as constant terms, for instance, PK_1 .

6.2.1 Predicates

Relation predicates can be expressed on int-typed attributes and comprise the predicates Eq , Neq , Leq , Lt , Gt , and Geq , standing for equality, non-equality and the inequalities, respectively, analogous as done for $\mathcal{L}^{P/C}$. Because the only attribute type in \mathcal{L}_{LL} is int, the predicate signature need not be made explicit. Whenever a relation predicate over an attribute of an object is expressed, it implies also holdership of the respective object, without this being made explicit.

A predicate $IdfRelProof$ can be expressed over a Pedersen or group commitment object p_j representing holdership of it without making any statements over the attributes as shown in (6.2).

$$IdfRelProof(p_j, h_j, s_j, sid_j) \quad (6.2)$$

The above predicate represents a proof of holdership of a Pedersen or group commitment of the commitment object p_j without revealing information on the committed values. Additionally, it associates the terms referred to in the $\mathcal{L}^{P/C}$ formula specified in the predicate with the identifier object. Technically, this relates the terms to the private key or private keys p_j is based on, that is, a private key of the party or a private key and delegation private key, from the (delegation) private key domains corresponding to the (delegation) registration domains the corresponding identifier object is member of.

A public key specification predicate $PubKey(o, PK)$ expresses that public key PK is to be used for the cryptographic protocol related to cryptographic object o . Through this, we specify the public keys for making cryptographic proofs of knowledge of signatures of private certificates or computing and proving correctness of Camenisch–Shoup verifiable encryptions. The key material required for computing Damgård–Fujisaki commitments is derived from other key material in a subsequent transformation and need thus not be specified.

Conditions related to conditionally released identities are related to the corresponding cryptographic objects through simple predicates, analogous to public keys.

A formula in \mathcal{L}_{LL} has a structure of sub-formulae corresponding to that of formulae in $\mathcal{L}^{P/C}$.

6.2.2 Transformation $\mathcal{M}_{\mathcal{L}_{LL}}$

The transformation $\mathcal{M}_{\mathcal{L}_{LL}}$ from a data statement $\phi \in \mathcal{L}^{P/C}$ to a formula $\phi_{LL} \in \mathcal{L}_{LL}$ translates certain identity semantics of $\mathcal{L}^{P/C}$ to semantics that can be expressed in a language talking about cryptographic objects. For each private certificate,

the corresponding certificate structure must be obtained for expressing relevant technical specifications, such as the mapping \mathcal{M} of attributes. Also instantiated DFO-commitment objects and ciphertext objects have a corresponding structure that needs to be available.

Our discussion of $\mathcal{M}_{\mathcal{L}_{LL}}$ starts with the mapping of the preamble of ϕ . The third-party specification sub-formula, identity introduction sub-formula, and conditionally released identity introduction sub-formula of ϕ are used for obtaining a sequence of public key specification predicates $\text{PubKey}(\mathbf{o}_i, \text{PK}_i)$, thereby realizing the certifier or conditional recipient semantics of primary identities and conditionally released identities of ϕ , as specified through the association of objects with third parties through the *certifier* and *recipient* attributes, respectively. The public key for a private certificate is determined through the *certifier* and *type* attributes of the corresponding identity. Furthermore, the conditionally released identity introduction subformula is mapped to predicates associating Camenisch-Shoup ciphertexts with release conditions.

The identifier object introduction subformula is mapped to a corresponding sub-formula making statements about the Pedersen or group commitment objects corresponding to the identifier objects. The predicates, which are defined for each primary identifier object \mathbf{p}_j which ϕ introduces, that associate the *subject*, *holder*, and *subjectId* attributes with constants through logical equality, are mapped to a predicate IdfRelProof , exposing the terms $\mathbf{h}_j, \mathbf{s}_j$, and sid_j . This formally associates the terms of $\mathcal{L}^{\text{P/c}}$ with the commitment.

Note that the terms used for the *holder* and *subject* of identifier objects are those resulting from the renaming specified through the nested equivalence relations \mathfrak{E} and \mathfrak{E}'_γ discussed in Sec. 4.8.2. Logical equality relations expressed between *holder* or *subject* constant terms of different identifier objects in ϕ are mapped to \wedge -connected equality relations $\text{Eq}(\mathbf{p}_i[k_i], \mathbf{p}_j[k_j])$ between the corresponding private key attributes with index k_i and k_j of the corresponding commitment objects with the meaning that equality of those private keys be proven in the cryptographic protocol.

A logical equality relation between two terms in the identifier object introduction sub-formula in ϕ for bridging registration domains is mapped to equality relations of the private key attributes of the corresponding domain bridging private certificate and the integers corresponding to the private keys of the commitment objects corresponding to the identifier objects of the bridged registration domains. This implies also that a proof of knowledge of the domain bridging certificate be specified, that is, predicates being added relating the corresponding private key attributes.

The mapping of the assertion sub-formula is the remaining functionality of $\mathcal{M}_{\mathcal{L}_{LL}}$. It transforms the assertion sub-formula of ϕ to an assertion sub-formula expressed over the cryptographic objects over only int attributes.

In this sub-formula, predicates with and without a $\langle \rangle$ annotation may be contained. When mapping these predicates to predicates of ϕ_{LL} , those annotations are retained structurally, that is, all target predicates obtained from mapping a predicate of ϕ_{LL} comprise a $\langle \rangle$ annotation if and only if the source predicate does. The annotation has implications on how the certificate structure of a private certificate is obtained, as well as on the cryptographic material, and governs whether crypto-

graphic protocols are to be proven based on secrets known by the prover or whether simulated transcripts are to be generated.

The mapping comprises the encoding of attributes of objects into `int`-typed attributes of the cryptographic objects, as governed through the function \mathcal{M} of the corresponding structure. An `int`-typed attribute of $\mathcal{L}^{p/c}$ is encoded as the corresponding `int` attribute of ϕ_{LL} while enforcing the restriction that the to-be-mapped integer be representable through its limited-length `int`-typed encoding. A `date` attribute is mapped to a single attribute of type `int`, also with a check of the encoding result being within the value domain of `int`. The complication hereby is that a date requires a 0-point as governed through the structure corresponding to the cryptographic object it is contained in, which can vary for each private certificate type, opaque identity type, or conditionally released identity type. The 0-point determines the date/time value which is encoded as the integer 0. A `str`-typed attribute is mapped to one or multiple attributes of type `int`, as governed through the structure of the cryptographic object the attribute is to be encoded in. An unlimited-length integer is encoded as a hash resulting from application of a collision resistant hash function to the attribute. This is the only attribute mapping option which does not encode the attribute value, that is, is not preserving the attribute value. A `bool` or `idf` attribute is mapped in a straightforward way to an `int` attribute. Further encodings which may be more efficient, though, require additional building blocks [CG08], are not included in our discussion now. It is not hard to integrate those into our generic framework and architecture. The indices of the target attribute for encoding an identity type are specified in the object structure.

Relation predicates are mapped one-by-one, such that the connective structure of ϕ is preserved. Additional parantheses may be added, if required for specifying the precedence, around a sub-formula obtained from mapping a single predicate.

A relation predicate with a given signature in ϕ expressed on two arguments is mapped to the corresponding predicate on `int`-typed arguments in \mathcal{L}_{LL} . An argument is mapped considering the above type mapping for the different data types and, in case of the argument being a generic `int` or `date` expression, also considering the complications incurred from the expression, that is, relating to multiple attribute references. For date expressions, additional complexity results from the 0-point encoding of the referred-to date attributes and also date granularities.

A predicate relating the *holder* or *subject* attribute of an identity with a constant `pty`-typed term requires special handling. Such a predicate is mapped to an `Eq` predicate between the corresponding private key attribute of the private certificate as determined through its structure and the private key attributes of the commitment object that exposes the respective `pty`-typed constant term in its corresponding `IdfRelProof` predicate. This cryptographically realizes the proof that those objects are based on the same private key or delegation private key. Private key attributes of commitment objects or private certificates are never related to constant terms through predicates, only to other such attributes, because an equality relation with a constant would reveal the private key.⁵⁹

⁵⁹The system implementation, the private key being contained in a tamper-resistant hardware

A feature predicate is mapped to a predicate representing the feature in \mathcal{L}_{LL} —we do not give further details here as this is not particularly involved or interesting.

The structure of the generated assertion sub-formula of ϕ_{LL} is equivalent to that of the source sub-formula of ϕ . With structural equivalence of the formulae we mean that, while a predicate of ϕ can result in a sequence of predicates in ϕ_{LL} , when considering such sequence as a single element of the formula, the formulae have the same logical connective structure. This is ensured through the construction and by not leaving the paradigm of ϕ_{LL} still being declarative.

6.3 Abstract Protocol Language

The *Abstract Protocol Language* \mathcal{APL} abstractly specifies cryptographic protocols through basic cryptographic building blocks, or primitives. Contrary to formulae of $\mathcal{L}^{P/c}$ and \mathcal{L}_{LL} , a formula expressed in \mathcal{APL} specifies a cryptographic protocol over the cryptographic objects instead of declaratively specifying relations among attributes.

6.3.1 Language \mathcal{APL}

The language is a formula-style language expressing predicates over elements, where a predicate corresponds to a cryptographic building block. A formula specifies the composition of building blocks to a composed cryptographic protocol for proving a data statement ϕ . The building blocks are abstract in \mathcal{APL} and specified in detail only in the next-lower layer in the stack using the language \mathcal{CPL} .

A cryptographic protocol is realized through composing instances of basic cryptographic protocols. \mathcal{APL} represents those instances of basic cryptographic protocols abstractly, without giving their detailed specification. Each basic cryptographic protocol that we use for composing a protocol is represented through at least one \mathcal{APL} predicate. Thereby, we distinguish between *compute predicates* and *prove predicates*. A compute predicate only comprises computation steps a party needs to perform locally, while a prove predicate represents a ZK-PoK to be executed with the other party in the protocol.

A basic cryptographic protocol is represented through predicates in a way that its non-prove-related protocol steps of the parties are represented as compute predicates for the prover or verifier, while the ZK-PoK part is represented through a single prove predicate used by both prover and verifier in their processing. For example, the proof of knowledge protocol of an SRSA-CL signature over a tuple of attributes [CL02b] is represented as compute protocols SRSACLrand or SRSACLsim for the prover, depending on whether the party needs to compute a proof or create a simulation transcript, SRSACLrec for the verifier, and a ZK-PoK specification SRSACLprove for both. Thereby, SRSACLrand randomizes the SRSA-CL signature, while SRSACLsim , in the simulation case, creates a tuple indistinguishable from

token, or combinations thereof, should prevent the private key from being revealed.

a signature, over which a simulated proof of knowledge transcript is to be computed. SRSACLrec captures the receiving of values computed by the prover through SRSACLrand or SRSACLsim and assigning them to variables of the cryptographic program expressed in $\mathcal{CP}\mathcal{L}$. SRSACLprove corresponds to the ZK-PoK of the randomized signature, thereby capturing both the protocol endpoints of the prover and the verifier.

Predicates are associated with terms corresponding to the cryptographic objects, terms corresponding to public keys, or terms resembling constants. Through the reuse of the same terms in different predicates, the building blocks of one instance of a basic protocol are related to each other, and so are different instances of (different) protocols. For example, a prover uses the predicate in (6.3) with the term c referring to the corresponding private certificate and PK referring to the corresponding verification public key and k to the number of attributes of the certificate. Later in the $\mathcal{AP}\mathcal{L}$ formula, the prover uses (6.4) with the same parameters and an additional set D of the released attributes of the certificate for relating those predicates to belong to the same protocol instance of the protocol for proving knowledge of an SRSA-CL signature. The same approach is used to relate different instances of (different) basic protocols with each other.

$$\text{SRSACLrand}(c, \text{PK}, k) \tag{6.3}$$

$$\text{SRSACLprove}(c, D, \text{PK}, k) \tag{6.4}$$

An appropriate representation through protocol fragments expressed in $\mathcal{CP}\mathcal{L}$ and their corresponding $\mathcal{AP}\mathcal{L}$ abstractions is devised, as outlined for the SRSA-CL signature prove protocol, for each basic cryptographic protocol required for our system. This comprises the optional BL-CL signature proof protocol [CL04], a generalization, for supporting all inequality relations, of Boudot's inequality proof over committed integers [Bou00], a computation of a Camenisch-Shoup ciphertext and verifiable encryption proof thereof [CS03], computation and proof of knowledge of DF commitments [DF02], computation and proof of knowledge of Pedersen and generalized group commitments, and proving linear relations between committed attributes [CS97b]. While equality of attributes is expressed through a linear relation, non-equality requires a separate protocol. The k -show protocol [CHK⁺06] for cloning protection and prove protocol that an element is contained in a dynamic accumulator [CL02a] realize the k -show per epoch feature and revocation feature for a private certificate, respectively.

A formula in $\mathcal{AP}\mathcal{L}$ expresses the composition of building blocks corresponding to instances of basic cryptographic protocols to the overall cryptographic protocol for proving the input formula.

6.3.2 Transformation $\mathcal{M}_{\mathcal{AP}\mathcal{L}}$

The mapping $\mathcal{M}_{\mathcal{AP}\mathcal{L}}$ implements the declarative semantics of a formula in \mathcal{L}_{LL} in a cryptographic protocol in $\mathcal{AP}\mathcal{L}$ using the building blocks for the basic cryptographic

protocols. Due to the paradigm change from \mathcal{L}_{LL} to \mathcal{APL} , the mapping $\mathcal{M}_{\mathcal{APL}}$ is substantially more involved than $\mathcal{M}_{\mathcal{L}_{LL}}$ and we can only present the basic idea of this mapping in the current summary chapter. $\mathcal{M}_{\mathcal{APL}}$ changes the representation paradigm from the formula being a declarative identity statement to it being a cryptographic protocol specification with execution semantics. In the transformation of ϕ_{LL} to \mathcal{APL} , the constraints of the basic cryptographic protocols, particularly the constraints relating to ZK-PoKs of linear relations between preimages of homomorphic functions, need to be enforced. The main effects this has is the introduction of new cryptographic objects, so called *representative objects* or *proxy objects* and the expression of protocols over those instead of the objects they proxy, and a related structural change of the part of the formula that corresponds to the assertion sub-formula of ϕ_{LL} .

Some cryptographic objects referred to by terms already in ϕ_{LL} need to be generated, or *instantiated*. This holds for to-be-generated DFO commitments and Camenisch-Shoup ciphertexts and for the optional generation of group commitments, corresponding to to-be-generated opaque and conditionally released identities and pseudonyms, respectively. Their attributes have been computed already with an earlier mapping and passed as additional input to $\mathcal{M}_{\mathcal{APL}}$. The computation is expressed through the appropriately parameterized compute predicates corresponding to the cryptographic protocol required.

Without giving the details, a DF commitment as proxy object is generated for attributes of cryptographic objects and a relation predicate in ϕ_{LL} over attributes of cryptographic objects is expressed over the proxy instead of the attribute itself. This simplifies processing and is required in the general case of formulae with disjunctions to be able to implement the relation semantics of \mathcal{L}_{LL} with cryptographic protocols. This concerns linear relations among preimages of the proofs of knowledge over homomorphic functions for realizing the relation predicates in the formula ϕ_{LL} in the general monotonous formulae.

For each uninstantiated Camenisch-Shoup ciphertext object referenced in ϕ_{LL} , a computation predicate for computing the ciphertext according to the $\langle \rangle$ annotations in ϕ_{LL} and the semantics is issued. For every ciphertext, a predicate for computing a DFO commitment of the attribute tuple of the ciphertext is generated as well as a predicate for proving knowledge of it. Equality predicates between the attributes of the ciphertexts and the DFO commitments are generated as well to relate them to each other. Analogous processing is applied for each DFO commitment referenced in ϕ_{LL} . This processing is performed only once for a formula and creates ciphertexts and commitments as well as proxy objects for those. That is, a proof of knowledge predicate for a ciphertext or commitment is done once per formula. Objects referred to only in non- $\langle \rangle$ -annotated predicates in ϕ_{LL} are created with random or otherwise suitably determined attributes. Note that the generation of the objects as outlined is specific to the prover in the interaction.

For processing the predicates in an \wedge -node of ϕ_{LL} , the following is done for a relation predicate: For simplicity of the processing, a predicate for a proof of knowledge of an SRSA-CL signature, or for creation of a proof simulation by annotating it with the $\langle \rangle^{\text{sim}}$ annotation, is added for each reference of an attribute of the cer-

tificate, though, only once per \wedge -node. This can be optimized in a post-processing step by removing not-required proxy objects. For each referenced attribute, a predicate for generating a proxy DFO commitment is placed, and also a predicate for proving holdership of it and relating the proxied attribute to that of the certificate proof. This creates proxy objects for certificate attributes in the current scope of the formula. The relation predicate on expressions over attribute references of any of the cryptographic objects is translated to an expression predicate expressed over the attributes of the proxy objects. Feature predicates are mapped to such of \mathcal{APL} in a straightforward manner. Non-availability of a $\langle \rangle$ annotation for a predicate in the source formula is translated to a $\langle \rangle^{\text{sim}}$ annotation of the generated predicates in ϕ_{APL} and availability of a $\langle \rangle$ annotation leads to creation of no annotation for the target predicates. That is, the $\langle \rangle^{\text{sim}}$ annotation in \mathcal{CPL} is the annotation with inverse semantics to the $\langle \rangle$ annotation in \mathcal{APL} .

Note that this approach may create multiple proof-of-knowledge predicates for a single certificate reference in different disjunctive branches of the formula. Different proof-of-knowledge protocol instances or simulations thereof for one certificate require that the same randomized certificate be used as homomorphic preimage of the homomorphic preimage proof of knowledge in order to implement the strict equality semantics that all references of an identity in ϕ refer to the same cryptographic object. This is subtle, though, crucial for generating a correct protocol.

6.3.3 Protocol Optimization

\mathcal{APL} is designed to support optimizations of the protocol in terms of obtaining more efficient cryptographic protocols with the same identity semantics. Such optimizations operate on the formula and change its structure, e.g., introduce additional proxy objects and change the corresponding predicates and whole subformulae. They are defined through patterns specifying optimizable subformulae and how they are transformed. Through the simplified processing for the transformation to \mathcal{APL} , performing basic optimization steps is mandatory for obtaining a reasonable protocol and avoiding the overhead introduced with the use of proxy objects.

Once an \mathcal{APL} formula ϕ_{APL} has been obtained, a formula that implements the same data release semantics can be constructed through executing an optimizing post-processing stage. Proxy objects can be removed, unless required, in such a post-processing optimization transformation. Also, multiple attributes (of one object) can be merged into a single proxy object in case this can reduce the computational effort. Further optimization transformations may reduce the number of SRSA-CL proof of knowledge predicates required in the formula. Such transformations may move the proof predicate up the tree representing the connective structure of the formula, introduce new proxy objects for its attributes, and express the original relation predicates over the attributes of the new proxy objects. Depending on the considered scope of such transformation, it requires to operate on a subtree and make substantial changes in it. Such optimization transformations for efficiency improvements can be complex and are best applied by following patterns and heuristics.

The capability of optimizing protocols shows the power of the \mathcal{APL} language and is one motivation for our run-time generation approach and its multi-layered architecture.

6.4 Cryptographic Protocol Language

\mathcal{CPL} , the *Cryptographic Protocol Language*, is used for specifying cryptographic building blocks realizing the basic cryptographic protocols. A protocol specification of a composed protocol expressed in \mathcal{CPL} results from transforming the representation of the protocol expressed in \mathcal{APL} using the transformation $\mathcal{M}_{\mathcal{CPL}}$. Each \mathcal{APL} predicate has a corresponding \mathcal{CPL} -specified cryptographic building block comprising the implementation details for the predicate required for the transformation $\mathcal{M}_{\mathcal{CPL}}$.

6.4.1 Languages \mathcal{CPL} and \mathcal{CPL}^*

For \mathcal{CPL} , we follow a *hybrid language design*, that is, deviate from standard language design of following a single language paradigm. \mathcal{CPL} comprises language elements with procedural and such with declarative semantics. A protocol expressed in \mathcal{CPL} is a sequence of statements having sequential execution semantics. A statement is either a *procedural*, or imperative, statement or a *declarative*, or ZK-PoK, statement. Due to the declarative statements, a protocol specification in \mathcal{CPL} cannot be executed by a procedural interpreter.

A procedural statement is one realizing imperative language structures such as loops or conditionals, or an atomic statement for executing an (algebraic) operation or declaring a type, that is, it is an instruction.

A declarative statement specifies a zero-knowledge proof of knowledge composed from possibly multiple proof parts, where each part is an instance of a basic cryptographic protocol. A declarative statement is expressed in an extension of the widely used Camenisch-Stadler notation [CS97a], where the latter is rather a specification of a proof goal than of the proof itself. Our extension is sufficiently concrete and explicit to serve as a proof specification for unambiguously specifying the proof protocol. Zero-knowledge proof of knowledge protocols are expressed over homomorphic functions defined in procedural operations.

The language referred to as *Executable Cryptographic Protocol Language*, or \mathcal{CPL}^* , is the fragment of \mathcal{CPL} without its declarative component and expresses executable cryptographic protocols. Thus, both a protocol specification in \mathcal{CPL} and a \mathcal{CPL}^* program for such specification can express the same cryptographic protocols at different abstraction levels, while only a \mathcal{CPL}^* program is executable by a reasonably simple protocol interpreter.⁶⁰

A specification in \mathcal{CPL} is expressed over group elements, e.g., such part of the cryptographic elements part of the cryptographic objects referred to in formulae in

⁶⁰A protocol interpreter with an integrated zero-knowledge proof-of-knowledge compiler can execute \mathcal{CPL} specifications.

\mathcal{L}_{LL} , referred to through constants and variables, much like constants and variables in a high-level programming language. For example, the cryptographic elements of an SRSA-CL signature or public key [CL02b] are referred to as group elements. All group elements are typed with the mathematical group they belong to through the $::^T$ -operator and groups are defined through separate operations. Homomorphic functions are defined with the function definition operator $::^F$. Procedural operations in a \mathcal{CPL} specification specify algebraic operations to be executed to assign values to variables, e.g., addition, multiplication, or multi-base exponentiation in a given group, much like the operations in a high-level programming language. Declarative statements also refer to variables and constants representing cryptographic values, homomorphic functions, or other entities, and specify zero-knowledge proofs of knowledge over homomorphic functions over those values.

6.4.2 Cryptographic Building Blocks

Each basic cryptographic protocol is represented through a set of *cryptographic building blocks*. Each such cryptographic building block is specified in detail using \mathcal{CPL} and a non- \mathcal{CPL} interface. A cryptographic building block comprises an *interface* and a *body*. The interface specifies input elements to the building block, elements exposed by the building block, dependencies in terms of other building blocks, and secret preimages the building block specifies proof-of-knowledge protocols over.

The *interface* comprises multiple sections, each of which are sketched next. The inputs comprise identifiers of typed elements being expected as *input* to an instance of the building block. *Exposed elements* are identifiers of typed elements that an instance of the building block exposes to a composed protocol. A *dependency* expresses that an instance of the current building block requires an instance of each building block listed among the dependencies. This is required for realizing the representation of a single protocol through multiple building blocks, that is, for “connecting” the building block instances for a protocol instance. The *preimage relations* specify the indices of preimages appearing as preimages in ZK-PoKs over homomorphic functions in a prove operation of the building block. Those preimages can be related through predicates to those of other building block instances of the composed protocol during protocol composition for realizing linear relations on attributes. All other preimages remain internal to the ZK-PoK operations of the building block.

The *body* is a fragment of a cryptographic protocol specified in \mathcal{CPL} . Statements in the body can refer to input elements, elements of dependencies, and elements declared within the body. A declared element can be exposed to other building blocks, that is, be accessible in a composed protocol from other building blocks. Those that are not exposed have the body of this building block as scope and are thus not visible outside of the building block.

A building block specifies a template of a fragment of a cryptographic protocol. The interface specifies aspects for the composition of an instance of the building block with instances of other building blocks. During transformation of an \mathcal{APL}

formula ϕ_{APL} to a CPL formula ϕ_{CPL} , building blocks corresponding to APL predicates are instantiated and composed. In the protocol composition, building blocks behave like macros when being instantiated and composed with other building blocks in that they are contextualized and expanded into a wider context.

6.4.3 Transformation \mathcal{M}_{CPL}

Mapping an APL formula through the transformation \mathcal{M}_{CPL} to a CPL specification uses, for each APL predicate, an instance of a CPL building block corresponding to an APL predicate and composes it with the specifications Γ and Δ for the procedural and declarative part of the specification, respectively, generated so far. An instance of a building block thereby is obtained by instantiating a building block, which generates a copy of the building block template and transforms this copy. The transformation steps comprise, among other steps, determining the input elements to the building block, resolving building block dependencies, and renaming elements in the body to refer to those of the inputs and building block dependencies, and renaming internal elements based on a protocol-wide unique identifier of the building block to separate the namespace of the instance of the building block from other instances. This realizes an integration of the building block, represented as a macro-like structure, through instantiation, into the context of the composed protocol. An instantiated building block is composed with both Γ and Δ , the composed protocol specifications comprising procedural and declarative operations, respectively.

The approach of building blocks acting as templates from which cryptographic protocol fragments for integration into the protocol to be composed are derived through instantiation allows for having sub-protocols specified in CPL independently and composing them to a single CPL specification for a given APL input. Through the macro approach, namespaces of building blocks are not perfectly isolated towards each other once composed, though, when assuming that the authoring of all building blocks used in a system is under control of one entity, this does not constitute a problem. Using a procedure-like representation of building blocks would not result in the explicit protocol representations we can obtain and potential drawbacks for their optimized execution for exploiting the inherent ILP.

The details of \mathcal{M}_{CPL} for instantiation and composition of building blocks are involved and not explained in this summary due to space constraints.

6.4.4 Transformation $\mathcal{M}_{\text{CPL}^*}$ to CPL^*

A composed protocol expressed in CPL still comprises declarative ZK-PoK operations which need to be compiled to procedural sub-programs for obtaining an executable cryptographic protocol specification, or (cryptographic) program, expressed in CPL^* . The mapping $\mathcal{M}_{\text{CPL}^*}$ from CPL to CPL^* is the final transformation step in our run-time protocol generation stack and outputs an executable *cryptographic program*. The mapping compiles the declarative zero-knowledge proof specifications to procedural sub-programs using the approach of zero-knowledge proof compilers.

A declarative \mathcal{CPL} operation specifies a Σ -protocol [Cra96], or a part thereof, where a Σ -protocol is, informally, a 3-round interactive protocol between a prover and a verifier, where the prover sends a commitment of randomness in the first round, the verifier a random challenge in the second round, the prover a response in the third round, and the verifier accepts if a verification equation on the protocol transcript can be verified. Σ -protocols are zero-knowledge proofs of knowledge. Note that such protocols can be made non-interactive using the Fiat-Shamir heuristics.

A zero-knowledge proof system informally is an interactive protocol through which an assertion can be proven without revealing any further information. Zero-knowledge proofs have been introduced by Goldwasser, Micali, and Rackoff [GMR85, GMR89]. Zero-knowledge proofs are a powerful concept in that all of \mathbb{NP} admits a zero-knowledge proof system [GMW86, GMW91] and even everything provable, that is, all of \mathbb{IP} is provable in zero knowledge as shown by Ben-Or et al. [BOGG⁺88]. Though, practically efficient⁶¹ proof systems exist only for a much more constrained class of languages, for which efficient protocols have been devised.

A *proof of knowledge* is, intuitively speaking, an interactive protocol through which a party can prove knowledge of an element x for a pair $(x, y) \in R$ being member of a relation R . Formalizing that a party knows something is non-trivial and has been done by Bellare and Goldreich [BG92] through the notion of a *knowledge extractor*.

6.4.5 Proof Protocols

The class of protocols of interest to us are the Σ -protocols, which are characterized the properties of special soundness and special honest verifier zero knowledge. Thereby, the protocols are zero-knowledge proof (or argument) systems with a knowledge error that is exponentially small in the challenge length [CDM00].

The concrete Σ -protocols of interest to us are proof of knowledge protocols of the preimages of homomorphic functions. The functions we consider are exponentiations or multi-base exponentiations, also referred to as multi-exponentiations, over different classes of groups, that is, homomorphic functions over those groups or product groups thereover.

The simplest class of protocols are proof of knowledge protocols for discrete logarithms over groups with finite domains [Ped91] or tuples of logarithms over product groups of such [CS97a]. The first known protocol instance for the further class is the Schnorr protocol [Sch91], for the latter the representation proof protocol by Camenisch and Stadler [CS97a]. This class of protocols is referred to as Σ^ϕ -protocols in the framework of Bangerter et al. [BKS⁺09]. Proof systems for those classes of protocols are zero-knowledge *proofs* of knowledge in the definition of Bellare and Goldreich [BG92].

Another important class of protocols are proof of knowledge protocols for preimages of exponentiation or multi-exponentiation homomorphisms in hidden-order

⁶¹The constructions used in [GMW86, GMW91, BOGG⁺88] are efficient in the sense of the term as it is used in theoretical computer science, that is, as being of polynomial complexity with respect to a parameter. Thus, efficient protocols in this notion may well be entirely impractical.

groups or product groups of such. The basic ideas for this have been put forth by Fujisaki and Okamoto [FO97], with Damgård and Fujisaki correcting the results [DF02]. The results have been generalized by Bangerter et al. [BCM05] and Bangerter [Ban05]. These protocols are *arguments* of knowledge in the definition of Bellare and Goldreich [BG92].

For an *argument of knowledge* [BG92], or computational proof of knowledge, soundness does not hold for a computationally unbounded prover—such prover may be able to convince a verifier about the truth of an assertion although it does not hold. In this thesis, we may use the term “proof” loosely, that is, for both proofs and arguments of knowledge in the sense of Bellare and Goldreich, unless the differentiation is of importance.

Camenisch et al. [CKY09] have devised a framework for zero-knowledge proofs for (multi-)exponentiation homomorphisms in (product) groups of known or hidden order. Bangerter et al. [BKS⁺09] have proposed an extension thereof achieving lower round complexity. The proofs of the latter are set in the auxiliary string model or common reference string model by Blum et al. [BFM88] requiring access of all parties to a common reference string, while the further does not build on this assumption. Overall, the main constructions of both frameworks are very similar.

Linear relations between homomorphic preimages of homomorphisms in a composed proof protocol can be handled as proposed in the literature [CS97a, Bra97, CS97b]. Essentially, a preimage relation can be seen as a proof over a product group using the standard proof protocol in the composition frameworks [CKY09].

A ZK-PoK specification can relate predicates expressed therein through conjunctions and disjunctions using a generic structure of connectives. Preimages may be related through linear relations—most commonly equality—with the constraint that only preimages directly comprised in the same conjunction node may be related. This relates to the constraints imposed on \mathcal{APL} on what can be realized with cryptographic protocols. When viewing preimage proofs of knowledge over homomorphic functions with related preimages as preimage proofs over corresponding product groups, this observation follows immediately.

For realizing the multi-exponentiation homomorphism protocols of the above-discussed frameworks, proofs that an integer lies in a given interval [CM99] are required as auxiliary cryptographic mechanisms.

Compiling a \mathcal{CPL} formula to \mathcal{CPL}^* can build on the zero-knowledge proof frameworks of Camenisch et al. [CKY09] or Bangerter et al. [BKS⁺09]. The latter is more suitable due to their 3-move protocols in which a proof can, without restrictions, be transformed to a signature of knowledge using the Fiat-Shamir approach [FS86].

The programmatic composition of cryptographic building blocks corresponds to a composition of Σ -protocols of the respective class using either of the above-discussed frameworks. The frameworks mandate some overhead in order to ensure soundness of the protocols, essentially the introduction of further DFO-type commitments over so-called safeguard groups in certain cases [CKY09, BKS⁺09].

A Σ -type composed protocol can either be executed as an interactive protocol, thereby obtaining the zero-knowledge property, or by creating a *signature of knowledge* as introduced by Camenisch and Stadler [CS97a] and later formalized by

Chase and Lysyanskaya [CL06] in a non-interactive way through the Fiat-Shamir approach [FS86] of using a (practical instantiation of a) random oracle [BR93] in place of the verifier and thereby obtaining a signature of knowledge. Only in the case of a signature of knowledge, a third-party-verifiable data statement can be obtained, which may be of interest for verification by a conditional recipient before decrypting Camenisch-Shoup ciphertexts and is crucial for the application to Fiat-Shamir-based PKI discussed earlier for it being a signature scheme.

6.4.5.1 Monotonous Formulae

Monotonous formulae, that is, formulae comprising disjunctions, can be realized using the approach of proofs of partial knowledge by Cramer, Damgård, and Schoenmakers [CDS94]. Predicates with a $\langle \rangle^{\text{sim}}$ annotation thereby are such for which the prover does not know the secret preimages, while for non- $\langle \rangle^{\text{sim}}$ -annotated predicates it does know them. The connective structure over the formula gives rise to an access structure which can be realized through a secret sharing scheme. Intuitively speaking, the approach by Cramer, Damgård, and Schoenmakers uses a suitable secret-sharing scheme for “splitting” a received challenge into sub-challenges for the different preimages, while allowing the prover to run a protocol simulator for the sub-protocols corresponding to the $\langle \rangle^{\text{sim}}$ -annotated predicates and only being required to compute the protocol for the non- $\langle \rangle^{\text{sim}}$ -annotated predicates. Thereby, the prover remains capable of correctly computing the third protocol round for a received challenge. The properties of the secret sharing scheme and the construction ensure soundness, that is, the property that the prover needs to know preimages corresponding to a qualified access structure, that is, needs to be able to prove all non- $\langle \rangle^{\text{sim}}$ -annotated predicates in our construction.

According to a theorem of Cramer, Damgård, and Schoenmakers [CDS94], using a *smooth* secret sharing scheme allows for obtaining a witness indistinguishable protocol [FS90] for the input being honest-verifier-zero-knowledge (HVZK) protocols with special soundness. Shamir’s secret sharing scheme is an example for a smooth scheme, though, is a threshold scheme and thus not applicable for realizing general access structures, unless one reverts to disjunctive normal form of the formula to be proven as done in relevant related work [BKS⁺09, BBH⁺09]. The general secret sharing scheme of Benaloh and Leichter [BL88] that is of interest for realizing general connective structures is only a *semi-smooth* scheme, for which reason this theorem of Cramer, Damgård, and Schoenmakers [CDS94] does not apply. They have proven a further theorem for obtaining a witness indistinguishable protocol from special HVZK protocols with special soundness property for semi-smooth secret sharing schemes. This theorem relaxes the smoothness property to semi-smoothness at the cost of additionally requiring that the protocols to be composed have the stronger special soundness property.

Witness indistinguishability (WI) as a protocol property has been introduced by Feige and Shamir [FS90] as the property that the verifier cannot distinguish which witness has been used by the prover, in case multiple witnesses could have been used. The witness indistinguishability property of the proof of partial knowledge ensures

that the prover leaks no information about which preimages the prover knows and thus which predicates are $\langle \rangle^{\text{sim}}$ -annotated. No statements on the zero-knowledge property of the composed protocol can be made for this construction.

An additional construction by Cramer et al. [CDS94] allows for obtaining witness hiding protocols, which is still a weaker property than the zero-knowledge property, though, may be sufficient for applications such as identification schemes [CDS94]. Cramer has proposed an approach for obtaining a perfect zero-knowledge proof system, also based on a semi-smooth secret sharing scheme and special HVZK protocols with the special soundness property, in his PhD thesis [Cra96], which has been revisited by Cramer, Damgård and MacKenzie [CDM00]. This construction builds on the above-discussed construction yielding witness indistinguishable protocols as a primitive and transforms a witness indistinguishable protocol into a perfect zero-knowledge protocol.

Though, Cramer et al.'s construction for obtaining perfect zero-knowledge protocols poses practical problems in our system as it requires that all relations over which the composed proof is expressed be known to the verifier upfront and that the verifier compute and prove holdership of a commitment for each of the relations as a first protocol step. Realizing this for our authentication flows where a party first obtains a policy and cryptographic challenge for executing a signature of knowledge protocol and responds with a signature of knowledge would require a higher round complexity than for a Fiat-Shamir-based signature of knowledge, where only an initial challenge is required from the verifier to ensure freshness. Concretely, the verifier would need to be notified by the prover of all relations the proof protocol makes use of before computing the proof.

6.4.6 Technical Realization

A zero-knowledge proof compiler transforming formal proof specifications similar to the declarative operations expressed in a \mathcal{CPL} formula into proof protocols has been designed and implemented in a related line of work [BBK⁺09, ABB⁺10, BBH⁺09, BKS⁺09, BABB⁺12]. A high-level overview of the concept of zero-knowledge proof generation and compiler-based implementation can be found in Bangerter et al. [BBK⁺09]. More recent work by this team proposes a certifying compiler [ABB⁺10, BABB⁺12], creating formal-methods-based proofs as assurance.

Using the above-discussed frameworks and approaches for composing Σ -protocols, one can compile the abstract ZK-PoK specifications of $\phi_{\text{CPL}} \in \mathcal{CPL}$ into a concrete protocol $\phi_{\text{CPL}^*} \in \mathcal{CPL}^*$. The available work on zero-knowledge proof compilers is a solid starting point. Some extensions required to integrate such compiler into our run-time protocol generation approach for realizing the mapping $\mathcal{M}_{\mathcal{CPL}^*}$ are discussed next.

The line of work on zero-knowledge proof compilers [BBK⁺09, ABB⁺10, BBH⁺09, BKS⁺09, BABB⁺12] always transforms monotonous formulae to disjunctive normal form for applying the construction of Cramer, Damgård, and Schoenmakers [CDS94] based on Shamir's secret sharing scheme [Sha79]. Depending on the connective structure of the formula, transforming to disjunctive normal form

may be sufficiently efficient for some applications. An important extension to the existing compilers is to build on a secret sharing scheme for general access structures and using Cramer et al.’s approach for realizing monotonous formulae as discussed above without reverting to disjunctive normal form. Without such extension, the compilers are still applicable when always transforming ZK-PoK statements of \mathcal{CPL} into disjunctive normal form.

A required extension concerns the output format of the proof compiler. Currently, Bangerter et al.’s work outputs protocol specifications in \LaTeX or the high-level programming language JAVA. A program in JAVA code is not suitable for our purposes, because a run-time-generated protocol would need to be compiled to JAVA byte code that can be executed by a JAVA virtual machine. This introduces computational and latency overhead and also does not allow for optimizations by exploiting the ILP inherent to the generated protocols in an as effective way as with protocols represented as programs in \mathcal{CPL}^* .

What is required for our framework is an output in \mathcal{CPL}^* or a similar language that can be translated to it. Then, a compiled protocol can be composed with the remaining cryptographic protocol expressed in \mathcal{CPL}^* .

6.4.7 Composability of Cryptographic Protocols

The approach for protocol composition of the cryptographic protocols we use, and related work building on non-trivial composed zero-knowledge proof systems alike, suffer, although using well-known approaches for protocol composition, from the problem that no formal proof of correctness of the overall composed cryptographic protocols exists—only for the individual building blocks. Doing such formal proof, either in the ideal-world/real-world simulation or game-based paradigm, is open work for our system. Particularly, the genericity of the protocols, resulting from them being a function of the input data statement ϕ , poses substantial challenges hereto.

Regarding zero-knowledge proofs, due to the properties of black-box simulation of the cryptographic protocols, concurrent composition poses the problem that a protocol simulator has runtime exponential in the number of concurrently executed protocols.

Those composability problems can be resolved by employing what is referred to as *universally composable* protocols. Universally composable primitives guarantee, when being composed with each other to a compound protocol and when polynomially many concurrent protocol instances being executed, their security properties, and thus avoid problems of composition of protocols from basic building blocks and concurrent composition. The cryptographic research community is active on universally composable protocols and promising results have become available [CKS11].

Our overall run-time protocol generation approach can be equally applied to universally composable protocols once efficient variants thereof will be available for the kinds of protocols we need.

6.5 Protocol Execution

A cryptographic program expressed in \mathcal{CPL}^* is executed by the respective party having generated it against the program generated by its protocol participant. Execution is realized by an optimizing interpreter exploiting the ILP inherent to the cryptographic programs through parallel execution of independent instructions. This is analogous to the approach a reduced instruction set computing (RISC) central processing unit (CPU) takes for exploiting the instruction-level parallelism (ILP) of assembly programs for those architectures based on, e.g., out-of-order issuing of instructions to multiple concurrent execution units [HP12].

The technical approach proposed by us is the construction of a dependency graph of the full \mathcal{CPL}^* program and an execution of the program by issuing independent instructions concurrently, governed by the dependency graph. Instructions in our setting are—in contrast to RISC assembly instructions—complex instructions, e.g., group operations in finite groups, such as multi-base exponentiation, requiring a huge number of (RISC) machine instructions to be executed on a modern processor. The concept of ILP applies analogously, though, with the assumption of an asynchronous execution machine for the instructions and different instruction execution times. It is easy to see from the individual basic cryptographic protocols that the ILP of composed protocols is substantial, thus resulting—when being exploited—in a considerable speedup of protocol execution.

The performance improvements obtained through exploitation of ILP during protocol execution and through optimization of \mathcal{APL} formulae are orthogonal to the performance optimizations that the zero-knowledge proof compiler of Bangerter et al. [BKS⁺09] can achieve. Implementation architectures which do not construct a cryptographic program and execute the cryptographic operations in-line with the processing of an input formula cannot exploit ILP of the protocol to the same extent.

Complementary to the abovementioned performance improvements are improvements on the arithmetic layer. A first simple improvement is using multi-base exponentiation using known algorithms instead of computing multi-exponentiations as product of powers.

Using well-known techniques for precomputing certain group elements to reduce the number of group operations for a multi-base exponentiation composes with efficiency improvements mentioned earlier. The cryptographic protocols we discuss require that (multi-)exponentiations with the same (tuples of) bases be done repeatedly, e.g., when using the same public key. Verifiers of zero-knowledge proofs can use precomputations for tuples of fixed bases with one variable base.

Operations with such structure can be sped up by precomputing group elements related to the fixed bases and reusing them throughout multiple (multi-)exponentiations with the same base tuples to obtain a space-time tradeoff for the arithmetic operations [Pip76, BGMW92, LL94, Som04a]. Those efficiency improvements for (multi-)exponentiations build on the computation of short addition chains or vector addition chains.

Those issues have been investigated by the author already in his master thesis [Som04a] in terms of efficient multi-exponentiation algorithms and a system

design and implementation for run-time profiling the (multi-)exponentiation operations being executed and performing precomputations to speed up the execution of frequently used base tuples.

6.6 Future Work

A complete implementation of our framework is left to future work, only a considerably simplified version thereof realizing a relatively small fragment of $\mathcal{L}^{p/c}$ has been implemented by the author based on a novel 2-layer architecture [Som07]. Such implementation would leverage the existing work on compilers for Σ -protocols and extend them as discussed. Integration of such compiler into our framework requires a mapping from \mathcal{CPL} to the language used for the compilers.

A detailed specification of our run-time protocol generation framework, of which this chapter is a summary, is planned to be published in the future.

For real-world deployments, an important part is the optimizing capability of the protocol interpreter for reducing protocol latency. Exploiting the IPL through out-of-order issuing and concurrently computing a large number of protocols as is the case for a verifier benefits the modular exponentiation throughput when using Graphics Processing Units (GPUs) for performing arithmetic operations. Therefore, it may be worthwhile to explore the application of GPUs in settings of servers performing a large number of verifications of credential proofs concurrently. Also, consideration of using special-purpose hardware for performing group operations may be important for practical deployments for being able to handle the computational load for verifying protocols on the server side.

Chapter 7

Implementation

A part of the research for this thesis comprises the implementation of selected aspects of our results for validating our approach. The implementation results are related to creating a system for privacy-enhancing identity management based on credential protocols. The implementation efforts have already been summarized in Sec. 1.5.1.2 in the introduction—next we present further technical information for selected efforts, though, without going into all the details.

7.1 Cryptographic Protocol Library

In this thesis we have summarized a generic and thus very powerful approach to the run-time generation of cryptographic protocols from a logic-based input specification in Chapter 6. The resulting protocol framework is based on a 5-layer processing stack and capable of generating protocols for realizing the semantics of data statements expressed in \mathcal{L}^P .

The author has implemented an earlier—and simpler—framework for generating and executing a simpler variant of the cryptographic protocols in the form of a cryptographic library [Som07]. The supported language is based on the language presented by the author in earlier work [BCS05, CSZ06a]. The language supports predicates over multiple credentials in a single proof protocol, verifiable encryption for realizing conditional release, and commitments for realizing opaque identities.

A major missing functionality of the library is the support for statements comprising disjunctions—it is a main contribution of this thesis to handle the relevant aspects of supporting disjunctions in formulae to be released for building a system. Considering the functionality, the library [Som07] realizes the data release protocols for an already strong subset of our approach for supporting anonymous, yet accountable, access control [BCS05] and our framework for privacy-enhanced PKI [CSZ06b, CSZ06a].

The library is based on a substantially simpler 2-layer architecture than that proposed in this thesis. The *high-level* layer translates an input formula for which a cryptographic protocol is to be generated into a low-level specification language, similar to the mapping $\mathcal{M}_{\mathcal{L}_{LL}}$ in the framework of Chapter 6. The similarities consist in the mapping being from a language expressed on identity management concepts to a language expressed over cryptographic objects, and go far in terms of technical similarities. That is, identity semantics is translated to semantics of cryptographic protocols.

The *low-level layer* translates a formula expressed in the low-level specification language into executable cryptographic protocols and executes those. The mapping is realized by an interpreter which maps predicates of the formula in the specification language to cryptographic building blocks, links different building blocks with each other as required for realizing the formula, and computes the cryptographic sub-protocols corresponding to the building blocks. Thus, this layer essentially comprises two sub-layers, one for mapping to the cryptographic building blocks and linking them with other building blocks, and one for executing the cryptographic operations corresponding to the building blocks. For this reason, one can argue this library being based on a 3-layer architecture.

The library realizes the arithmetic layer in a way that possibilities for optimizations for the group arithmetic, such as multi-base exponentiations, are accounted for in the technical architecture. The author has addressed efficient arithmetic for cryptographic systems in his master thesis [Som04a]. This work has encompassed a study of multi-base exponentiation, specialized algorithms for considering the specific structure of credential protocols, an approach for trading off memory against runtime by run-time profiling of cryptographic operations being executed, and an implementation of all this in a system based on the JAVA language framework.

The library is, to the best of our knowledge, the first implementation of the cryptographic protocols for privacy-enhanced authentication based on a multi-layered approach for transforming a high-level input to cryptographic protocols. It is the successor of an earlier implementation of a simpler library for the credential protocols by the author done in early 2002 which also remained internal to IBM. The library is, to the best of our knowledge, the first implementation that builds upon using the idea of certificate structures for specifying structural information of cryptographic certificates required for executing the mapping from the high-level input formulae to the cryptographic protocols.

The approach summarized in Chapter 6 is more generic than the above-discussed library and supports a more expressive specification language. The additional mapping layer $\mathcal{M}_{\mathcal{APL}}$ allows for protocol optimizations, while the generation of

an executable program expressed in the language \mathcal{CPL}^* allows for exploiting the instruction-level parallelism of the generated cryptographic protocols through parallel execution.

7.2 WS-Trust Request-Response Flows

Execution of a cryptographic 2-party protocol like the privacy-enhanced authentication protocols based on credentials requires messages being exchanged between the parties. For real-world systems, standards issued by industry groups exist on how to represent such messages and what their semantics is. Examples for such standards are the WS-Security [WSS04] and WS-Trust [KN03a] standards. WS-Security is based on XML-DSIG [ERS02] for digital signatures and WS-Trust builds on WS-Security.

The author has proposed a method that utilizes the verifiable pseudo-random function of Dodis and Yampolskiy [DY05] for generating a temporary signature key based on an identity statement and zero-knowledge proof of its correctness and a proof in zero knowledge that the key pair has been generated properly through the verifiable random function [CGS06]. Using this temporary key, the prover of a 2-party protocol can bind the identity statement using the standard semantics of WS-Trust [KN03a], while not requiring an extension of the underlying XML digital signature standard [ERS02], which was perceived as crucial at the time. This enables a new WS-Federation Active Requestor profile [KN03b] based on private certificates instead of conventional tokens.

We have shown the integration of the request-response message flow with the WS-Trust industry standard. The flow comprises a user obtaining a policy and random challenge from a service provider, determining a data statement fulfilling the policy and computing a cryptographic proof for it [Som07], and sending the statement and proof to the service provider for verification. The integration of the messaging with the WS-Trust standard has been reported in detail in Camenisch et al. [CGS06] implementing the private certificate protocols. The integration with the standard requires cryptographic operations to be executed which have been realized as a separate library for a clean modeling from a software engineering perspective.

7.3 User Interface

We have implemented a simple yet powerful user interface allowing a user to choose how to fulfill a data request of another party, as reported in Camenisch, Shelat et al. [CSSZ06]. The user interface has been realized as a Web browser extension for Firefox which renders the interface directly within the browser window the user interacts with. The interface is rendered co-located to the interface element on the Web page, clicking of which triggers the identity management protocol flow. The advantage of such approach is that there is a strong visual link between the resource to access through the interface element representing it on the Web page and the user

interface for identity selection. It particularly also allows a user to open multiple identity selections at once in different browser windows, e.g., to compare policies of different service offerings.

The browser extension is constrained to only rendering data requests (policies) and allowing a user to select how to fulfill the policies and communicating with the locally executed process implementing policy matching and the cryptographic protocols for obtaining the policy files and letting this process know about the selection. That is, the use interface component does not implement substantial processing logic. This is left to the back-end process, which is implemented using JAVA and realizes the functionality for policy matching and performing the cryptographic proof based on the data statement the user selects.

As technologies we chose the open-source Firefox Web browser due to its flexible support of extensions and the XML User Interface Language (XUL), an XML-based language for implementing user interfaces, for the implementation of the browser-based user interface.

7.4 End-to-end Privacy-preserving Access Control

For demonstrating the feasibility of our approach of privacy-preserving access control, we have designed and implemented a fully functional end-to-end prototype thereof. The prototype realizes a scenario in the healthcare domain. The basic idea is that an insurance company offering healthcare insurance services provides its members access to a healthcare portal with health-related information as an extra service. The system allows users to log into an access-protected healthcare information portal on a subscription basis without revealing identifying attributes. Thereby, it differentiates different types of credentials and authorizations that can be obtained through releasing parts of the attribute information of the credentials: A user can either present a health insurance credential giving access to a basic part of the portal, or an additional IBM employee credential for proving being a corporate member, which gives access to a wider selection of resources. Those options were formalized in the simple authorization policies deployed in the system.

This system shows the use of a single as well as multiple credentials to gain access to resources. Thereby, predicates for revealing partial information contained in the credentials as well as predicates for relating non-released attributes of the credentials, e.g., equality predicates over the private keys the credentials are based on, have been realized through cryptographic protocols.

Technically, the server-side system was based on an instance of the Tomcat application server, the authentication system of which was used to accept data-minimizing proofs instead of username/password pairs as authentication tokens and, depending on the attributes give access to respective sets of Web resources. The cryptographic library of the author [Som07] with its integration into the WS-Trust standard [CGS06] was used as authentication and messaging mechanism.

The user-side system features a Firefox Web browser with a plug-in that is triggered upon the user clicking on an access control policy and invokes a user-

side back-end process. This process matches the obtained policy with the user's credentials, queries the user for selecting a fulfilling statement of her choice, and computes a cryptographic proof based on private certificates of the user.

The prototype implements earlier variants of the results presented in this thesis [BCS05, CSZ06b, CSZ06a], our user interface work [CSSZ06], and the results on integration of the message flow with WS-Trust [CGS06]. Using an extension of the underlying technical architecture, the results of this thesis can be integrated with standards and, based on this, with commercial products, in the same way. The more general access control system and negotiation protocol proposed by the author [Som11] will require a multi-round message flow and additional processing, though based on the same architecture.

The full system was presented at the Internet Identity Workshop 2007 in the Computer History Museum in Mountain View, California [BSBC⁺07]. The system realizes the complete protocol flow for data-minimizing access to an online resource. The user thereby accesses an access-protected resource through her Web browser. The response of such request is a data request provided to the user's software system. The software system matches the request against the user's portfolio of private certificates. The results of the matching are displayed to the user for her selection which determines with which possible combination of attributes of her suitable certificates she intends to fulfill the request.

7.5 Ontology-based Reasoning

In the scope of our work on using ontologies for reasoning on authorization [HS06], a prototypical implementation of the concepts was built. This implementation was capable of computing whether a data statement fulfills a data request of a policy using reasoning over an ontology. The data representation language used was RDF [ME04], the ontology language OWL [MvHE04]. Data statements were to be proven by a predecessor of our credential library [Som07]. Automated reasoning was realized with a JAVA-based open-source reasoner by Hogben.

Building on a fragment of description logic, the prototype was among the first to use the approach of using named nested graphs for representing disjunctions. This work was performed as part of the efforts related to a comprehensive system for user-centric privacy-enhancing identity management created as one of the technical contributions of the PRIME project [PRI08].

Chapter 8

Conclusions

We have designed a system and framework for privacy-enhancing identity management in open settings such as that of the Internet. The system allows users to authenticate to service providers while minimizing the revealed data to what is required by the service provider. The system supports the *outsourcing of trust* decisions, e.g., regarding certification properties of attributes, similar to the delegation of trust as proposed in trust management systems. The system supports *delegation of attribute authority*, which is a crucial requirement of practical identity management systems. It allows for *binding attributes to parties* in a flexible way through multiple related registrations of a user instead of assuming a single registration. As a particularly important feature, it supports the possibility of having accountability of requesters authenticated through a non-identifying attribute statement, that is, achieving *accountability and privacy* at the same time. Results on *user interfaces* for selecting data statements to be released for fulfilling a policy have been obtained. The integration of the messaging related to obtaining a policy and sending a proof message with the *WS-Trust standard* and also with products building on this standard has been shown. We have shown furthermore how the required *cryptographic protocols* for establishing identifier objects, establishing identities, and releasing data based on those integrate with each other and the complete system, particularly the *logic-based data model*. We have developed an approach for the run-time generation of cryptographic protocols and summarize it in the thesis. An *abstract authentication model* for expressing the capabilities of our authentication system has been

presented. We have shown how the cryptographic protocols can realize relevant transformation rules of this model. Overall, the system and framework have been defined based on the Privacy-by-Design paradigm of considering privacy right from the beginning.

Only jointly employing multiple disciplines and extending and combining approaches from logic, access control, cryptography, and other domains, while considering usability, available standards, and system constraints to obtain an integrated system, made our comprehensive results possible. This interdisciplinary and integrating technical approach was not only necessary—it has constituted one of the most exciting challenges of the work.

Our results have substantially advanced the state of the art in the privacy-enhancing authentication and identity management domain by taking an *integrated approach* to the topic and addressing privacy-enhancing identity management from theory to practice, meaning to start the work with cryptographic protocols which were available and building a system around those protocols to bring them to practice. The subtitle “From Cryptography to Practice” of the thesis has been chosen to reflect this approach.

A part of the results presented in this thesis has been presented in the form of research publications, which have been consolidated and integrated to the presented coherent system and framework. We have validated our results through *implementation of selected aspects* of earlier variants thereof. Thereby, we have obtained a fully functional prototype of a privacy-enhanced identity management system based on a healthcare scenario, showing the practical feasibility of the data-minimizing authentication aspects using cryptographic credential protocols. For further emphasizing practical viability of privacy-enhancing authentication protocols, we have integrated the protocol flow for accessing an online resource using messaging following the WS-Trust standard and have prototypically integrated this with a commercial product.

In addition to the privacy-enhanced identity management system and framework, we have made further contributions in related and orthogonal spaces as discussed in the introduction. Those cover a wider perspective of privacy related to electronic communication networks and data processing.

Due to the complexity of the Internet and Web ecosystem and personal data economy, our results related to the system and framework for privacy-enhanced identity management only provide solutions for parts of the privacy problem space. Orthogonal contributions we have made cover the important area of electronic social networks, which are a main cause of user data dissemination and related privacy problems. Thereby, such results are crucial for protecting users’ privacy. Also, we show how to integrate our privacy-enhanced identity management system with virtual reality platforms, thereby obtaining an architecture which reuses the real-world authentication infrastructure and maps it into the virtual world. Those orthogonal results increase coverage of the privacy-related problem space in today’s electronic communication systems. However, both in our core area of work as well as in further areas, a plethora of items are left for future work.

8.1 Open Problems

We next discuss open problems towards practical deployments of data-minimizing authentication technologies of the kind discussed in this thesis. We have already presented open issues in greater detail in multiple chapters of the thesis and summarize those aspects here, while extending our view to a more comprehensive coverage of the area of user privacy for electronic communication networks.

In the *trust management* area, an interesting branch of future work is combining the advantages of our ontology-based approach with the approach taken in trust management systems, leading to a more context-dependent use of ontologies depending on assurance requirements of a transaction. Discussions related to this have already started in an early report on the topic [HS06]. A generic approach towards trust management has many challenges on the non-technical side, including such in the areas of business models, liability aspects, and bootstrapping such system, which are also part of future work.

Regarding our approach of data modeling as presented in Chapter 4, an implementation of an efficient deduction engine with support for policy matching for the full proposed language is an open issue. Optionally, such deduction engine can operate on sufficiently expressive fragments of the data statement and request languages. A policy editor for authoring policies based on our data request language is also left for future work. Due to the complexity of the language, incurred mainly due to the privacy features, a simple presentation for the policy author seems required.

The *cryptographic protocols* we build upon require further research to be done regarding their composability, on the one hand composability of multiple cryptographic building blocks to a cryptographic protocol, and on the other hand concurrent composability of multiple such composed protocols. The so-called universal composability (UC) framework provides a widely accepted framework for composing cryptographic protocols in a strong model. Universally composable cryptographic protocols are an active field of research and a promising result on universally composable zero-knowledge proof of knowledge protocols has recently been presented by Camenisch et al. [CKS11], which suggests that strong composability of the kinds of Σ -protocols we are interested in does not incur a large overhead. This ongoing work in the cryptographic community will provide us with the theoretical foundation for future versions of the cryptographic protocols. Those results are expected to fit into our approach of run-time generating the cryptographic protocols from individual provably composable cryptographic building blocks for implementing a high-level protocol specification and will affect only the lowest layers concerned with the cryptographic building blocks of the processing stack we have defined.

Regarding *efficiency and latency* of the execution of cryptographic protocols, tailored algorithms and implementations thereof, including precomputation-based optimizations, can lead to lower CPU requirements and latency of protocols. For large-scale server-side deployments, using full-custom application-specific integrated circuits (ASICs) or the massive parallelism of graphics processing units (GPUs) for realizing the arithmetic for the protocols will be necessary for reducing the energy footprint and space and cooling requirements in datacenters for those computation-

ally expensive protocols. The “green” aspect, that is, the carbon footprint as well as datacenter-related costs of the cryptographic protocols we propose have not been prominent in discussions so far.

In the *usability* arena, we have obtained results on user interfaces for identity selection for data-minimizing cryptographic protocols [CSSZ06, BCPS09]. Considering those and the relevant work in the literature body, there is still a need for further work in this area. Particularly, when considering the more expressive language proposed in this thesis, compared to the languages the abovementioned results build on, additional challenges arise, e.g., data statements comprising disjunctions and the resulting data-minimizing semantics as well as the support for accountability features for data-minimizing transactions. A crucial success property of a privacy-enhancing identity management system is likely to be the non-intrusiveness of the system in terms of not interrupting the usual workflows of the users more than today’s approaches and being intuitive to use.

A practical system does, as argued in the introduction, not only require a privacy-enhancing authentication system as the one proposed in this thesis. It will require the *integration of multiple mechanisms* for protecting user privacy, such as policy-driven life-cycle data management, tracking control tools, and traffic-layer privacy protection mechanisms, all controlled in a concerted manner by a client system enforcing a user’s preferences.

We think that preferences-based *decision support systems* for enforcing a user’s intentions of data release and handling will be required for practical and usable identity management systems. Those can build on user-defined preferences and privacy metrics to support users in their identity management decisions. For a practical system, those support systems will need to control multiple privacy-enhancing technologies to balance a user’s privacy against data release for the purposes of service provisioning and customization and directed advertising. This is an area of active research, however, in our view, comprises a very hard-to-address problem domain. As is true for any user-side preferences-based system, the default preferences shipped by the vendors of the client system will be crucial for the obtained degree of protection and are out of scope of the technical discussions.

An interesting item of work related to the deployment of a system we propose comprises the question of *privacy-preserving directed advertising*, that is, directed advertising in the setting of using data-minimizing authentication consistently, while also limiting third-party tracking and employing techniques for reducing the release of information through the communication protocol. Particularly, it may be interesting to investigate cryptographic technologies for realizing privacy-enhanced directed advertising, that is, the involved third parties being able to provide targeted advertisements while learning less data required for the targeting than today. New mechanisms in this space may lead to a reshaping of the current data economy, through increasing privacy throughout the current ecosystem while retaining utility of the data. This may be crucial for practical acceptance of consistent use of data-minimizing solutions as it will reflect interests of the different players in the personal data economy. It is completely open whether reasonable mechanisms in this space will be available.

It is expected that *Big Data analytics* may lead to new challenges regarding user privacy due to the extended capabilities of knowledge inference from available data and the imbalance resulting therefrom. Data-minimizing authentication is an important building block in this context, however, it is not able to resolve the resulting challenges of large-scale data analytics.

A production-grade *implementation* of the proposed system is an important piece of open work. Our implementations are prototype implementations without fulfilling necessary criteria related to the software development, review, and certification process for a production-grade security software library. An open source implementation of a simpler private certificate system is currently being done as part of the European ABC4Trust government-funded program [ABC13], which includes the first real-world pilots of the technology.

An *integration with applications* of a data-minimizing identity management system for different classes of applications may be crucial for use cases not based on a Web browser. We have shown both the integration of protocol flows for credential protocols into a relevant standard [CGS06] as a basis for a browser integration [CSSZ06] and integration with an open source application server [BSBC⁺07]. At the time when virtual world systems were at, or close to, the peak of the hype cycle, we have also realized a prototypical integration of our authentication infrastructure with a major virtual world platform [BCH⁺09, BHSS10]. Analogously, such integration is required for any kind of application or class of applications to leverage the identity management system therein. This amounts to a pure implementation rather than a research task as the basic architectural principles have been worked on already, however, it is necessary for supporting a wide range of applications.

Because we focus on the technical aspects of privacy-enhanced identity management in this work, we are intentionally missing out on multiple relevant *non-technical aspects* for practical deployments of our technology or related technologies by businesses on the Internet. First and foremost, an aspect we do not consider is the required re-design of business processes, which is necessary to allow companies to retain the service quality offered today while operating on a reduced data set. This has many non-technical implications which need further work towards a real-world system deployment.

Furthermore, a crucial aspect is the investigation of *legal aspects* of data-minimizing interactions, particularly in the sense of compatibility of data-minimizing transactions with law enforcement requirements and national security. At least European data protection legislation and a recent initiative in the USA comprise data minimization as a main principle. Accountability of users in a setting of data-minimizing transactions is subject to be worked on in the legal research domain.

Automating a user's identity management decisions and addressing legal questions on the extent to which this is possible, e.g., in the European user consent regime [Eur95], are further future research questions of interest. This comprises not only automated identity selection, but also the presentation of policies to the user and obtaining user consent as well as the automated release of information based on user preferences.

Besides the abovementioned items of future work, there are further related areas

of interest we have not mentioned here, e.g., usage control or data handling, data anonymization, or privacy-preserving data mining.

Orthogonal to data-minimizing authentication are mechanisms for protecting user privacy at the application layer for certain classes of particularly privacy-sensitive applications provided to end users. The most prominent example for such applications are *electronic social networks* for which our work on access control to profile data in electronic social networks [MS12a] can mitigate part of the privacy issues. In the area of back-end data processing performed, for example, by service providers or government agencies, our result for privacy-enhanced *identity resolution* [VPCS11a, VPCS11b] provides a certain degree of protection of user privacy by allowing parties to share obfuscated data sets instead of plaintext data sets, while retaining utility of the data to a large degree and being practically efficient. Our results in those areas are a starting point, however, they do not provide an exhaustive solution to the issues. Further research in those and a plethora of other areas are required for addressing today's privacy issues in a comprehensive manner.

8.2 Outlook

The current trend of mobile computing and execution environments providing isolation properties between regular and trusted application environments, such as the Trusted Execution Environment (TEE) [Glo11] or so-called secure elements (SEs)—essentially smart cards—integrated with mobile devices, and particularly combinations thereof, may give rise to a ubiquitous deployment of platforms applicable for securely performing authentication transactions over communication networks in the near future. Such security-enhanced platforms can act as secure storage location for a user's private key(s) and certificates, can realize the user interface for secure input and output, and perform (parts of) the cryptographic protocols related to authentication transactions. Orthogonally or complementary to this, current ongoing developments in the domain of realizing privacy-enhancing protocols on future government-issued tokens, particularly electronic driver's licenses (eDLs) in the investigation of which the author is involved, may lead to authentication tokens applicable in a widespread spectrum of authentication use cases. Both upcoming mobile platforms and developments in the space of government-issued (identity) tokens can be seen as an ideal practical deployment platform for an authentication solution as the one discussed in this thesis. Deploying our system based on such end-user devices and tokens can resolve main architecture-related issues on how to ensure security of private keys and thus the strong binding of attributes to parties as well as protection against vulnerabilities related to the software stack of end user devices.

Current regulatory trends are clearly going towards strengthening privacy [Eur12, The12] in some of the world's regions, including the data minimization aspect. With the dynamic technology development in the area, it can be expected that the proposed data-minimizing authentication technologies increasingly become state of the art, thereby being increasingly mandated by regulations. This reflects

the usual life-cycle of technologies and their transition from research technologies to becoming state of the art and facing widespread deployment.

Complementary to the proposals for strengthening legislation in Europe and the USA, government-funded research projects have been supporting technology development. The European PRIME [PRI08, CLS11], PrimeLife [Pri10], and ABC4Trust [ABC13] research programs and the USA-based NSTIC [The11, Nat10] initiative are prime examples for this. While PRIME and PrimeLife had a strong focus on data-minimizing protocols and practical realization thereof and a broader view on privacy, such as social network privacy and data handling, respectively, the goal of ABC4Trust is to create an open source implementation of credential protocols and the related infrastructural components and to pilot them in a selected set of real-world use cases. NSTIC can be seen as both a complementary and competing initiative to the European programs that has been launched in the USA. The author has, as part of and in addition to his technical contributions in the context of his thesis, made major technical contributions to PRIME and managed the PrimeLife research program.

Frequent data breaches have been leading to a *change of attitude* within companies towards storing excessive amounts of personal data in certain industry sectors due to the substantial damages companies can incur in case of data breaches [Dow08a]. Technologies we propose can help companies push data minimization further compared to today's deployed technologies and mitigate the risks by contributing to a reduction of expected data breach costs. Orthogonal security measures can be employed to reduce breach risks and thus further lower the overall expected costs related to data breaches.

Thanks to the frequent privacy breaches and other privacy issues in areas directly affecting peoples' lives, such as in electronic social networks, *end user awareness* has risen in the recent years. This is clearly reflected by the frequent coverage in mainstream press and media in general. Particularly, the financial disadvantage and inconvenience resulting from identity theft related to data breaches is increasingly perceived by the citizens.

Data-minimizing authentication has gained traction also in the field of *standards*, e.g., proposals in the scope of the European Citizen Card [Eur13] related to data-minimizing mechanisms have been made, however, those relate to weaker mechanisms than the ones proposed in this thesis. A recently started initiative with major involvement by the author investigates data-minimizing identity management, considering private certificate technology, in the field of ISO-compliant electronic driver's licenses [Int09] within a joint working group of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Benefits of a potential future privacy-enhanced electronic driver's license are not only a greater utility for use cases in the driving domain due to additional protection features and the capability of it to be used in online interactions. Mainly, such driver's licenses could, due to the potential widespread roll out, become identity tokens applicable for secondary use cases and thereby become an integral part of a future identity ecosystem building on privacy-enhanced protocols. Considering the standardization activities of the recent years, privacy has clearly

gained substantial attention in international standardization [ISO13, ISO11]. Also in the domain of industry standards, multiple initiatives related to privacy can be observed. In the author's opinion, certain international standards or industry standards will be crucial for a successful large-scale deployment of the technologies we propose. Standards are one means of making mechanisms state of the art through the widespread acceptance and trust in a technology and consensus regarding details thereof that a widely agreed standard brings about. Once mechanisms are considered to be state-of-the-art mechanisms, businesses may be legally required to use them.

Considering the above trends in the different areas of relevance clearly shows that privacy and data minimization have gained substantial traction in the recent years within the technical community, companies, standardization bodies, regulators, governments, and society at large. This can be seen as an indication of an increasing readiness of the technical, economic, government, and societal environment for data-minimizing authentication and therefore a possible deployment of such technologies in the mid-term future.

In order to achieve the openness of our system, different kinds of *players* are needed besides service providers relying on statements being authenticated and users employing the system for authenticating statements. Certifiers need to register users, optionally verify their legal identities out of band, and issue corresponding attribute certificates or issue assigned attributes. This role is expected to be fulfilled, at least partially, by government certifiers issuing identity tokens such as identity cards or driver's licenses supporting (a subset of) our cryptographic mechanisms. A completely new type of player are ontology providers who endorse statements about other parties and take liability for those statements. This is closely related to auditing parties and vouching for the assessed properties towards other players in the system. Players taking the role of conditional data recipients for the purpose of obtaining transactions with revocable anonymity are another kind of trustee service that needs to emerge. The full potential of the system we put forth in this thesis can only be leveraged with all those kinds of players being available in an identity ecosystem.

A major problem for a deployment of cryptographic credential technology is that establishing an *identity management ecosystem* comprising a plurality of the required kinds of parties is, due to interdependencies and the currently non-obvious revenue generation models, an inherently complex undertaking. Initially, individual parties in such ecosystem are not incentivized to make investments for deploying technology because monetary value can, due to the interdependencies, be generated only once a certain market penetration has been reached. We think that such ecosystem can be best bootstrapped by a number of government certifiers or mobile operators registering users and issuing private certificates with attributes of high assurance. Based on this, service providers can start supporting the technology and allow users to interact with them in a privacy-enhancing way. Once the created value of such initial system is perceived in the marketplace, the identity ecosystem can grow and the full potential of the technology can be leveraged. As common for paradigm-changing technologies, a multitude of opportunities for both established

and new businesses will arise in such an identity ecosystem.

Our results obtained in the context of this thesis are a major step forward in the area of system aspects of privacy-preserving identity management systems and thereby can contribute to better user privacy in the future. Taking those and the complementary and orthogonal results in the literature body into account, the author claims that the core technical challenges have been sufficiently resolved by now to not be considered the main obstacle to a first deployment of privacy-enhanced identity management, while substantial remaining challenges for establishing a sustainable identity ecosystem are mostly of non-technical nature and relate to the above.

In our opinion, one always needs to consider privacy-enhanced technologies of the kind we discuss in the wider context they are to be deployed in, otherwise crucial aspects related to their potential deployment and system characteristics are missing because of considering a subsystem without its connection to practice. At a first glance, today's multi-billion dollar *personal data economy* and the proposed privacy-enhanced technologies might seem like incompatible with each other. However, there are multiple possibilities of a coexistence of both, as discussed next.

An extreme scenario in terms of aggressive data minimization through performing all transactions with the maximum-possible degree of data minimization with respect to authenticated attribute statements and preventing all profiling is not realistic to happen in the near to mid term in the opinion of the author, because it would counter the personal data economy and substantial economic interests of multiple players in today's Internet ecosystem and innovation.

A realistic scenario, in the view of the author, is a scenario where the privacy interests of users are balanced with the interests of the commercial entities in the personal data economy.⁶² Authentications are performed in a data-minimizing way based on certified attributes, with the multilateral benefits of this as discussed earlier, while sufficient profiling information is obtained by service providers and optionally third parties to allow for customizing services and targeting advertisements for financing and driving the innovation of the overall Web ecosystem.

An extreme scenario in terms of data exploitation through a personal data economy is a scenario with extensive profiling as today for a large fraction of the interactions, however, with privacy-enhanced authentication and further privacy technologies being employed in a concerted manner as governed by the users for certain sensitive transactions. Our argumentation is that also in a data economy as today, there is a need for data-minimizing identity management technology, at least for a specific fraction of user interactions, to safeguard users' privacy interests.

A paradigm shift in terms of how a future personal data economy operates might be induced through the emergence of a *personal data market*, in which users receive monetary value in exchange for the right to use their personal data for purposes not related to transactions they execute. It can be argued that such a paradigm allows for much better user control over personal data disclosures and their use. Also in such a new paradigm, the technologies proposed in this thesis are crucial for helping

⁶²Balancing interests is in the tradition of the European value system, which is also reflected in the European data protection laws.

minimize expected data breach costs, for the availability of high-assurance attributes for use cases such as age or creditworthiness verification, and for minimizing data release in transactions, and thereby benefit users and service providers alike.

When looking back to the state of cryptographic protocols for user-centric privacy-enhanced authentication a decade and more ago and comparing the situation with today, the topic has become prominent in research, industry, and among policy makers, while hardware technology has progressed substantially to allow for short execution times of the cryptographic protocols. Considering this and the efforts currently going on, we can argue that the overall environment has become more favorable for a real-world deployment of the proposed advanced authentication technologies.

References

- [ABB⁺10] José Bacelar Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider, *A certifying compiler for zero-knowledge proofs of knowledge based on sigma-protocols*, ESORICS (Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, eds.), Lecture Notes in Computer Science, vol. 6345, Springer, 2010, pp. 151–167.
- [ABC13] *ABC4Trust project Web site*, 2013, <https://abc4trust.eu/>.
- [ACC⁺05] Christer Andersson, Jan Camenisch, Stephen Crane, Simone Fischer-Hübner, Ronald Leenes, Siani Pearson, John Sören Pettersson, and Dieter Sommer, *Trust in PRIME*, Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005 (ISSPIT 2005), 2005, pp. 552–559.
- [ACDS08] Claudio A. Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati, *A privacy-aware access control system*, Journal of Computer Security (JCS) **16** (2008), no. 4, 369–392.
- [ACK⁺10] Claudio Agostino Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio, *Exploiting cryptography for privacy-enhanced access control: A result of the PRIME project*, Journal of Computer Security **18** (2010), no. 1, 123–160.
- [AF99] Andrew W. Appel and Edward W. Felten, *Proof-carrying authentication*, ACM Conference on Computer and Communications Security (Juzar Motiwalla and Gene Tsudik, eds.), ACM, 1999, pp. 52–62.
- [AFHWP12] Julio Angulo, Simone Fischer-Hübner, Erik Wästlund, and Tobias Pulls, *Towards usable privacy policy display and management for PrimeLife*, Information Management & Computer Security **20** (2012), no. 1, 4–17.
- [AGBR13] Jan Philipp Albrecht, Marielle Gallo, Franoise Le Bail, and Kostas Rossoglou, *The European data protection framework under review*:

- The proposed Regulation*, Panel discussion at the Conference for Computeres, Privacy & Data Protection, 2013 (CPDP 2013), available at http://www.cpdpcferences.org/23012013_1030.html, 2013.
- [AM10] April Adams and Naveen Mishra, *User survey analysis: Key trends shaping the future of data center infrastructure through 2011*, Gartner Dataquest Research Note G00208112, October 22, 2010.
- [And09] Nate Anderson, “Anonymized” data really isn’t—And here’s why not, available at <http://arstechnica.com/tech-policy/2009/09/your-secrets-live-online-in-databases-of-ruin/>, September 8, 2009.
- [Asi05] Asia-Pacific Economic Cooperation (APEC), *APEC Privacy Framework*, Electronic Commerce Steering Group (ECSG) of the APEC, December 2005.
- [Asi12] ———, *Guidebook on APEC Privacy and Trustmark*, Tech. report, Electronic Commerce Steering Group (ECSG) of the APEC, November 2012.
- [BABB⁺12] José Bacelar Almeida, Manuel Barbosa, Endre Bangerter, Gilles Barthe, Stephan Krenn, and Santiago Zanella Béguelin, *Full proof cryptography: Verifiable compilation of efficient zero-knowledge protocols*, Proceedings of the 2012 ACM conference on Computer and communications security (New York, NY, USA), CCS ’12, ACM, 2012, pp. 488–500.
- [Ban05] Endre Bangerter, *Efficient zero knowledge proofs of knowledge for homomorphisms*, Ph.D. thesis, 2005.
- [BBC⁺09] Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Thomas Heydt-Benjamin, Dieter Sommer, and Gregory Zaverucha, *Cryptographic protocols of the Identity Mixer library*, Tech. Report IBM Zurich Research Report RZ3730, IBM Research – Zurich, March 19, 2009.
- [BBCS10] Carl Binding, Anthony Bussani, Jan Camenisch, and Dieter M. Sommer, *Interactive selection of identity information satisfying policy constraints*, United States Patent Application Publication, No. US 2010, 0100926 A1, April 22, 2010.
- [BBG⁺07] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael K. Reiter, *Consumable credentials in linear-logic-based access-control systems*, NDSS, The Internet Society, 2007.
- [BBH⁺09] Endre Bangerter, Thomas Briner, Wilko Henecka, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider, *Automatic generation*

- of Sigma-protocols*, EuroPKI (Fabio Martinelli and Bart Preneel, eds.), Lecture Notes in Computer Science, vol. 6391, Springer, 2009, pp. 67–82.
- [BBK⁺09] Endre Bangerter, Stefania Barzan, Stephan Krenn, Ahmad-Reza Sadeghi, Thomas Schneider, and Joe-Kai Tsay, *Bringing zero-knowledge proofs of knowledge to practice*, Security Protocols Workshop (Bruce Christianson, James A. Malcolm, Vashek Matyas, and Michael Roe, eds.), Lecture Notes in Computer Science, vol. 7028, Springer, 2009, pp. 51–62.
- [BCG⁺10] Anthony Bussani, Jan L. Camenisch, Thomas R. Groß, Dirk Husemann, and Dieter M. Sommer, *Confidential presentations in virtual world infrastructures*, United States Patent Application Publication, No. US 2010/0058443 A1, March 4, 2010.
- [BCH⁺09] Anthony Bussani, Jan L. Camenisch, Thomas R. Gross Dirk Husemann, Ansgar Schmidt, and Dieter Sommer, *Method, system, and computer program product for virtual world access control management*, United States Patent Application Publication, US 2009/0254968 A1, October 8, 2009.
- [BCL04] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya, *A cryptographic framework for the controlled release of certified data*, Security Protocols Workshop (Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, eds.), Lecture Notes in Computer Science, vol. 3957, Springer, 2004, pp. 20–42.
- [BCM05] Endre Bangerter, Jan Camenisch, and Ueli M. Maurer, *Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order*, Public Key Cryptography (Serge Vaudenay, ed.), Lecture Notes in Computer Science, vol. 3386, Springer, 2005, pp. 154–171.
- [BCPS09] Patrik Bichsel, Jan Camenisch, Franz-Stefan Preiss, and Dieter Sommer, *Dynamically-changing interface for interactive selection of information cards satisfying policy requirements*, Tech. Report RZ 3756, IBM Research – Zurich, 2009.
- [BCS05] Michael Backes, Jan Camenisch, and Dieter Sommer, *Anonymous yet accountable access control*, Proceedings of the 2005 ACM workshop on Privacy in the electronic society (New York, NY, USA), WPES '05, ACM, 2005, pp. 40–46.
- [BCS12a] Patrik Bichsel, Jan Camenisch, and Dieter Sommer, *A calculus for privacy-friendly authentication*, Proceedings of the 17th ACM symposium on Access Control Models and Technologies (New York, NY, USA), SACMAT '12, ACM, 2012, pp. 157–166.

- [BCS12b] Patrik Bichsel, Jan Camenisch, and Dieter Sommer, *A calculus for privacy-friendly authentication*, Tech. report, IBM Research – Zurich, 2012.
- [BF11] Liana B. Baker and Jim Finkle, *Sony PlayStation suffers massive data breach*, Reuters, <http://www.reuters.com/article/2011/04/26/us-sony-stoldendata-idUSTRE73P6WB20110426>, April 26, 2011.
- [BFG10] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon, *SecPAL: Design and semantics of a decentralized authorization language*, Journal of Computer Security **18** (2010), no. 4, 619–665.
- [BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis, *The KeyNote trust-management system, version 2*, IETF RFC 2704, September 1999.
- [BFK99] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis, *The role of trust management in distributed systems security*, Secure Internet Programming (Jan Vitek and Christian Damsgaard Jensen, eds.), Lecture Notes in Computer Science, vol. 1603, Springer, 1999, pp. 185–210.
- [BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy, *Decentralized trust management*, IEEE Symposium on Security and Privacy, IEEE Computer Society, 1996, pp. 164–173.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali, *Non-interactive zero-knowledge and its applications*, Proceedings of the twentieth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '88, ACM, 1988, pp. 103–112.
- [BFS98] Matt Blaze, Joan Feigenbaum, and Martin Strauss, *Compliance checking in the policymaker trust management system*, Financial Cryptography (Rafael Hirschfeld, ed.), Lecture Notes in Computer Science, vol. 1465, Springer, 1998, pp. 254–274.
- [BG92] Mihir Bellare and Oded Goldreich, *On defining proofs of knowledge*, CRYPTO (Ernest F. Brickell, ed.), Lecture Notes in Computer Science, vol. 740, Springer, 1992, pp. 390–420.
- [BGF06] Moritz Y. Becker, Andrew D. Gordon, , and Cédric Fournet, *SecPAL: Design and semantics of a decentralized authorization language*, Tech. Report no. MSR-TR-2006-120, Microsoft Research, September 2006.
- [BGMW92] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley, and David Bruce Wilson, *Fast exponentiation with precomputation (extended abstract)*, EUROCRYPT (Rainer A. Rueppel, ed.), Lecture Notes in Computer Science, vol. 658, Springer, 1992, pp. 200–207.

- [BHSS10] Anthony Bussani, Dirk Husemann, Ansgar Schmidt, and Dieter Sommer, *Method, system, and computer program product for providing e-token based access control for virtual world spaces*, United States Patent, No. US 8,132,235 B2, March 6, 2010.
- [Bic07] Patrik Bichsel, *Theft and misuse protection for anonymous credentials*, Master's thesis, November 2007, Available at: <ftp://ftp.tik.ee.ethz.ch/pub/students/2007-HS/MA-2007-42.pdf>.
- [BKS⁺09] Endre Bangerter, Stephan Krenn, Ahmad-Reza Sadeghi, Thomas Schneider, and Joe-Kai Tsay, *On the design and implementation of efficient zero-knowledge proofs of knowledge*, ECRYPT workshop on Software Performance Enhancements for Encryption and Decryption and Cryptographic Compilers (SPEED-CC'09), October 2009.
- [BL88] Josh Cohen Benaloh and Jerry Leichter, *Generalized secret sharing and monotone functions*, in Goldwasser [Gol90], pp. 27–35.
- [Bla11] Bob Blakley, *2012 planning guide: Identity and privacy*, Gartner Report, November 2011.
- [BLFM05] Tim Berners-Lee, Roy Fielding, and Larry Masinter, *Uniform Resource Identifier (URI): Generic syntax*, IETF RFC 3986, 2005.
- [BMP⁺09] Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicio, *Security and trust through electronic social network-based interactions*, CSE (Los Alamitos, CA, USA), vol. 4, IEEE Computer Society, 2009, pp. 1002–1007.
- [BMU08] Michael Backes, Matteo Maffei, and Dominique Unruh, *Zero-knowledge in the applied Pi-calculus and automated verification of the direct anonymous attestation protocol*, Proceedings of the 2008 IEEE Symposium on Security and Privacy (Washington, DC, USA), SP '08, IEEE Computer Society, 2008, pp. 202–215.
- [BOGG⁺88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway, *Everything provable is provable in zero-knowledge*, in Goldwasser [Gol90], pp. 37–56.
- [Bor97] Willem N. Borst, *Construction of engineering ontologies for knowledge sharing and reuse*, Ph.D. thesis, September 1997.
- [Bou00] Fabrice Boudot, *Efficient proofs that a committed number lies in an interval*, EUROCRYPT (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer, 2000, pp. 431–444.
- [BR93] Mihir Bellare and Phillip Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, ACM Conference on Computer and Communications Security (Dorothy E. Denning, Raymond

- Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, eds.), ACM, 1993, pp. 62–73.
- [Bra97] Stefan Brands, *Rapid demonstration of linear relations connected by boolean operators*, EUROCRYPT (Walter Fumy, ed.), Lecture Notes in Computer Science, vol. 1233, Springer, 1997, pp. 318–333.
- [Bra00] Stefan A. Brands, *Rethinking public key infrastructures and digital certificates: Building in privacy*, MIT Press, Cambridge, MA, USA, 2000.
- [BS02] Piero A. Bonatti and Pierangela Samarati, *A uniform framework for regulating service access and information release on the Web*, Journal of Computer Security **10** (2002), no. 3, 241–272.
- [BSBC⁺07] Abhilasha Bhargav-Spantzel, Anthony Bussani, Jan Camenisch, Thomas Groß, and Dieter Sommer, *Privacy-enhancing healthcare prototype based on anonymous credentials*, Presented at Internet Identity Workshop 2007 (IIW 2007), Maintain View, CA, USA, 2007.
- [BSCGS06] Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Groß, and Dieter Sommer, *User centrality: A taxonomy and open issues*, in Juels et al. [JWG06], pp. 1–10.
- [BSCGS07] ———, *User centrality: A taxonomy and open issues*, Journal of Computer Security **15** (2007), no. 5, 493–527.
- [Bus98] Samuel R. Buss, *An introduction to proof theory*, Elsevier Science B.V., 1998.
- [Cam03] Jan Camenisch, *Identity mixer protocol specification*, IBM internal protocol specification, March 2003.
- [Car10] Nicholas Carr, *Tracking is an assault on liberty, with real dangers*, The Wall Street Journal, <http://online.wsj.com/article/SB10001424052748703748904575411682714389888.html>, August 6, 2010.
- [Cas11] Carsten Casper, *Top five issues and research agenda, 2011 to 2012: The privacy officer*, Gartner Report, 2011.
- [Cas12] Claude Castelluccia, *Behavioural tracking on the Internet: A technical perspective*, European Data Protection: In Good Health? (Serge Gutwirth, Ronald Leenes, Paul De Hert, and Yves Pouillet, eds.), Springer, 2012.
- [CCS06] Marco Casassa Mont, Stefano Crosta, and Dieter Sommer, *PRIME Architecture V2*, PRIME project, 2006.

- [CD00] Jan Camenisch and Ivan Damgård, *Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes*, ASIACRYPT (Tatsuaki Okamoto, ed.), Lecture Notes in Computer Science, vol. 1976, Springer, 2000, pp. 331–345.
- [CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie, *Efficient zero-knowledge proofs of knowledge without intractability assumptions*, Public Key Cryptography (Hideki Imai and Yuliang Zheng, eds.), Lecture Notes in Computer Science, vol. 1751, Springer, 2000, pp. 354–373.
- [CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, in Desmedt [Des94], pp. 174–187.
- [CG08] Jan Camenisch and Thomas Groß, *Efficient attributes for anonymous credentials*, ACM Conference on Computer and Communications Security (Peng Ning, Paul F. Syverson, and Somesh Jha, eds.), ACM, 2008, pp. 345–356.
- [CGS06] Jan Camenisch, Thomas Groß, and Dieter Sommer, *Enhancing privacy of federated identity management protocols: anonymous credentials in WS-Security*, Workshop on Privacy in the Electronic Society (WPES) (Ari Juels and Marianne Winslett, eds.), ACM, 2006, pp. 67–72.
- [CH02] Jan Camenisch and Els Van Herreweghen, *Design and implementation of the Idemix anonymous credential system*, ACM Conference on Computer and Communications Security (Vijayalakshmi Atluri, ed.), ACM, 2002, pp. 21–30.
- [Cha81] David L. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the ACM **24** (1981), no. 2, 84–90.
- [Cha85] David Chaum, *Security without identification: transaction systems to make big brother obsolete*, Communications of the ACM **28** (1985), no. 10, 1030–1044.
- [Cha86] ———, *Showing credentials without identification. signatures transferred between unconditionally unlinkable pseudonyms*, Proceedings of a workshop on the theory and application of cryptographic techniques on Advances in cryptology – EUROCRYPT ’85 (New York, NY, USA), Springer-Verlag New York, Inc., 1986, pp. 241–244.
- [Cha90] ———, *Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms*, Advances in Cryptology AUSCRYPT ’90 (Jennifer Seberry and Josef Pieprzyk,

- eds.), Lecture Notes in Computer Science, vol. 453, Springer Berlin Heidelberg, 1990, pp. 245–264.
- [CHK⁺06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich, *How to win the clonewars: Efficient periodic n -times anonymous authentication*, ACM Conference on Computer and Communications Security 2006, ACM Press, 2006.
- [CKS11] Jan Camenisch, Stephan Krenn, and Victor Shoup, *A framework for practical universally composable zero-knowledge protocols*, ASIACRYPT (Dong Hoon Lee and Xiaoyun Wang, eds.), Lecture Notes in Computer Science, vol. 7073, Springer, 2011, pp. 449–467.
- [CKY09] Jan Camenisch, Aggelos Kiayias, and Moti Yung, *On the portability of generalized Schnorr proofs*, EUROCRYPT (Antoine Joux, ed.), Lecture Notes in Computer Science, vol. 5479, Springer, 2009, pp. 425–442.
- [CL01] Jan Camenisch and Anna Lysyanskaya, *An efficient system for non-transferable anonymous credentials with optional anonymity revocation*, EUROCRYPT (Birgit Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer, 2001, pp. 93–118.
- [CL02a] ———, *Dynamic accumulators and application to efficient revocation of anonymous credentials*, Advances in Cryptology – CRYPTO 2002, volume 2442 of LNCS, Springer Verlag, 2002, pp. 61–76.
- [CL02b] ———, *A signature scheme with efficient protocols*, Third Conference on Security in Communication Networks – SCN '02, Volume 2576 of LNCS, Springer Verlag, 2002, pp. 268–289.
- [CL04] ———, *Signature schemes and anonymous credentials from bilinear maps*, Advances in Cryptology – CRYPTO, Springer Verlag, 2004.
- [CL06] Melissa Chase and Anna Lysyanskaya, *On signatures of knowledge*, CRYPTO (Cynthia Dwork, ed.), Lecture Notes in Computer Science, vol. 4117, Springer, 2006, pp. 78–96.
- [CLHS11] Jan Camenisch, Ronald Leenes, Marit Hansen, and Jan Schallaböck, *An introduction to privacy-enhancing identity management*, Digital Privacy (Jan Camenisch, Ronald Leenes, and Dieter Sommer, eds.), vol. 6545, Springer Berlin Heidelberg, 2011, pp. 3–21.
- [CLS11] Jan Camenisch, Ronald Leenes, and Dieter Sommer (eds.), *Digital privacy – PRIME – Privacy and identity management for Europe*, Lecture Notes in Computer Science, vol. 6545, Springer, 2011.

- [CM99] Jan Camenisch and Markus Michels, *Proving in zero-knowledge that a number is the product of two safe primes*, EUROCRYPT (Jacques Stern, ed.), Lecture Notes in Computer Science, vol. 1592, Springer, 1999, pp. 107–122.
- [CMN⁺10] Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer, *A card requirements language enabling privacy-preserving access control*, SACMAT (James B. D. Joshi and Barbara Carminati, eds.), ACM, 2010, pp. 119–128.
- [CMS10] Jan Camenisch, Sebastian Mödersheim, and Dieter Sommer, *A formal model of Identity Mixer*, Proceedings of the 15th international conference on Formal methods for industrial critical systems (Berlin, Heidelberg), FMICS’10, Springer-Verlag, 2010, pp. 198–214.
- [Con96] United States Congress, *Health insurance portability and accountability act of 1996*, Public Law 104-191, 1996.
- [Con12] Consultative Committee of the Convention for the Protection of Individuals with Regard to Automatic processing of Personal Data, *Final document on the modernisation of Convention 108*, T-PD(2012)04 rev en, http://www.coe.int/t/dghl/standardsetting/dataprotection/TPD_documents/T-PD_2012_04_rev_en.pdf, September 2012.
- [Cou50] Council of Europe, *The European Convention on Human Rights*, November 1950.
- [Cou81] ———, *Convention for the protection of individuals with regard to automatic processing of personal data*, 1981.
- [Cou10] ———, *Council of Europe response to privacy challenges – modernisation of Convention 108*, 32nd International Conference of Data Protection and Privacy Commissioners, 27–29 October 2010, Jerusalem, Israel, 2010.
- [Cou13] Council of Europe Communications, *Uruguay becomes the first non-European state to accede to personal data protection “Convention 108”*, http://www.coe.int/t/dghl/standardsetting/dataprotection/News/Press-release-FINAL-Uruguay-revised_EN.pdf, April 2013.
- [CPS11] Stephen Crane, Siani Pearson, and Dieter Sommer, *Introduction: Privacy, trust, and identity management*, in Camenisch et al. [CLS11], pp. 141–149.
- [Cra96] Ronald Cramer, *Modular design of secure, yet practical cryptographic protocols*, Phd thesis, University of Amsterdam, 1996.

- [CS97a] Jan Camenisch and Markus Stadler, *Efficient group signature schemes for large groups (extended abstract)*, in Kaliski Jr. [Kal97], pp. 410–424.
- [CS97b] ———, *Proof systems for general statements about discrete logarithms*, Tech. Report 260, Institute for Theoretical Computer Science, ETH Zürich, 1997.
- [CS03] Jan Camenisch and Victor Shoup, *Practical verifiable encryption and decryption of discrete logarithms*, CRYPTO (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer, 2003, pp. 126–144.
- [CSS⁺05] Jan Camenisch, Abhi Shelat, Dieter Sommer, Simone Fischer-Hübner, Marit Hansen, Henry Krasemann, Gérard Lacoste, Ronald Leenes, and Jimmy C. Tseng, *Privacy and identity management for everyone*, Digital Identity Management (Vijay Atluri, Pierangela Samarati, and Atsuhiko Goto, eds.), ACM, 2005, pp. 20–27.
- [CSSZ06] Jan Camenisch, Abhi Shelat, Dieter Sommer, and Roger Zimmermann, *Securing user inputs for the web*, in Juels et al. [JWG06], pp. 33–44.
- [CSSZ07] Jan L. Camenisch, Abhi A. Shelat, Dieter M. Sommer, and Roger D. Zimmermann, *Method and computer system for performing transactions between a client and a server*, United States Patent Application Publication, No. US 2007/0288750 A1, December 13, 2007.
- [CSZ06a] Jan Camenisch, Dieter Sommer, and Roger Zimmermann, *A general certification framework with applications to privacy-enhancing certificate infrastructures*, SEC (Simone Fischer-Hübner, Kai Rannenberg, Louise Yngström, and Stefan Lindskog, eds.), IFIP, vol. 201, Springer, 2006, pp. 25–37.
- [CSZ06b] ———, *A general certification framework with applications to privacy-enhancing certificate infrastructures*, Tech. Report RZ7379, IBM Research – Zurich, 2006.
- [Des94] Yvo Desmedt (ed.), *Advances in cryptology - CRYPTO '94, 14th annual international cryptology conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, Lecture Notes in Computer Science, vol. 839, Springer, 1994.
- [Deu03] Bundesrepublik Deutschland, *Bundesdatenschutzgesetz (BDSG)*, BGBl. I, January 2003.
- [DF02] Ivan Damgård and Eiichiro Fujisaki, *A statistically-hiding integer commitment scheme based on groups with hidden order*, ASIACRYPT (Yuliang Zheng, ed.), Lecture Notes in Computer Science, vol. 2501, Springer, 2002, pp. 125–142.

- [DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson, *Tor: The second-generation onion router*, USENIX Security Symposium, USENIX, 2004, pp. 303–320.
- [Dow08a] Dow Jones and Company Inc., *New payment card data mantra is “Don’t need it, don’t store it”*, Wall Street Journal (September 16, 2008).
- [Dow08b] ———, *Data breaches surpass 2007 level, but businesses rarely are penalized*, Wall Street Journal (September 9, 2008).
- [DP60] Martin Davis and Hilary Putnam, *A computing procedure for quantification theory*, Journal of the ACM **7** (1960), no. 3, 201–215.
- [DY05] Yevgeniy Dodis and Aleksandr Yampolskiy, *A verifiable random function with short proofs and keys*, Proceedings of Public Key Cryptography (2005), vol. 3386 of LNCS, Springer, 2005, pp. 416–431.
- [Eck10] Peter Eckersley, *How unique is your Web browser?*, Proceedings of the 10th international conference on privacy-enhancing technologies (Berlin, Heidelberg), PETS ’10, Springer-Verlag, 2010, pp. 1–18.
- [ERS02] Donald E. Eastlake 3rd, Joseph M. Reagle, and David Solo, *XML-Signature syntax and processing*, <http://www.w3.org/TR/xmlsig-core/>, March 2002.
- [Eur95] European Parliament, *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data*, Official Journal L (1995), 31–50.
- [Eur02] ———, *Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)*, Official Journal L (2002).
- [Eur12] European Commission, *Proposal for a regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)*, January 2012.
- [Eur13] European Committee for Standardization (CEN), *CEN prTS 15480 Identification card systems – European Citizen Card*, CEN Standard, CEN/TC 224, WG 15, 2010–2013.
- [Flo13] Gary Flood, *Sony slapped with \$390,000 U.K. data breach fine*, InformationWeek (January 24, 2013).

- [FO97] Eiichiro Fujisaki and Tatsuaki Okamoto, *Statistical zero knowledge protocols to prove modular polynomial relations*, in Kaliski Jr. [Kal97], pp. 16–30.
- [FS86] Amos Fiat and Adi Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, CRYPTO (Andrew M. Odlyzko, ed.), Lecture Notes in Computer Science, vol. 263, Springer, 1986, pp. 186–194.
- [FS90] Uriel Feige and Adi Shamir, *Witness indistinguishable and witness hiding protocols*, STOC (Harriet Ortiz, ed.), ACM, 1990, pp. 416–426.
- [FW11] Geoffrey A. Fowler and Ben Worthen, *Hackers shift attacks to small firms*, The Wall Street Journal, <http://online.wsj.com/article/SB10001424052702304567604576454173706460768.html>, July 21, 2011.
- [Gar11] Gartner, *Gartner says half of all organizations will revise their privacy policies by end-2012*, <http://www.gartner.com/newsroom/id/1761414>, 2011.
- [GBB⁺06] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter, *A linear logic of authorization and knowledge*, ES-ORICS (Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, eds.), Lecture Notes in Computer Science, vol. 4189, Springer, 2006, pp. 297–312.
- [Gen35] G. Gentzen, *Untersuchungen über das logische Schliessen*, Mathematische Zeitschrift **39** (1935), 176–210, 405–431.
- [Gir90] Jean-Yves Girard, *Proofs and types*, Cambridge University Press, 1990.
- [Gla79] Dorothy J. Glancy, *The invention of the right to privacy*, Arizona Law Review **21** (1979).
- [Glo11] Global Platform Alliance, *TEE systems architecture v1.0*, GlobalPlatform Standard, <http://www.globalplatform.org/specificationsdevice.asp>, December 2011.
- [GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff, *The knowledge complexity of interactive proof-systems (extended abstract)*, STOC (Robert Sedgewick, ed.), ACM, 1985, pp. 291–304.
- [GMR89] ———, *The knowledge complexity of interactive proof systems*, SIAM Journal on Computing **18** (1989), no. 1, 186–208.

- [GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, IEEE Annual Symposium on Foundations of Computer Science (1986), 174–187.
- [GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, Journal of the ACM **38** (1991), no. 3, 690–728.
- [GN08a] Yuri Gurevich and Itay Neeman, *Distributed-knowledge authorization language – January 2008 revision*, Tech. Report January 2008 revision of Microsoft Research Technical Report MSR-TR-2007-116 of August 2007, Microsoft Research, 2008.
- [GN08b] ———, *DKAL: Distributed-Knowledge Authorization Language*, CSF, IEEE Computer Society, 2008, pp. 149–162.
- [Gol90] Shafi Goldwasser (ed.), *Advances in cryptology – CRYPTO ’88, 8th annual international cryptology conference, Santa Barbara, California, USA, August 21–25, 1988, proceedings*, Lecture Notes in Computer Science, vol. 403, Springer, 1990.
- [GOS04] Nicola Guarino, Daniel Oberle, and Steffen Staab, *What is an ontology?*, Handbook on Ontologies (Steffen Staab and Rudi Studer, eds.), International Handbooks on Information Systems, Springer, 2004.
- [Gre03] Graham Greenleaf, *APEC Privacy Principles: More Lite with every version*, Privacy Law & Policy Reporter **10** (2003).
- [Gre04] ———, *APECs privacy standard regaining strength*, Privacy Law & Policy Reporter **10** (2004).
- [Gre12a] ———, *Global data privacy laws: 89 countries, and accelerating*, Privacy Laws & Business International Report (2012).
- [Gre12b] ———, *The influence of European data privacy standards outside Europe: Implications for globalization of Convention 108*, International Data Privacy Law **2** (2012).
- [Gre12c] ———, *Strengthening and ‘modernising’ Council of Europe Data Privacy Convention 108*, Privacy Laws & Business International Report (2012).
- [Gru93] Thomas R. Gruber, *A translation approach to portable ontology specifications*, Knowledge Acquisition **5** (1993), 199–220.
- [GSC07] Thomas R. Groß, Dieter M. Sommer, and Jan L. Camenisch, *Assertion message signatures*, International Patent Application, No. PCT/IB2007/051546, November 29, 2007.

- [HBC⁺04] Marit Hansen, Peter Berlich, Jan Camenisch, Sebastian Clauß, Andreas Pfitzmann, and Michael Waidner, *Privacy-enhancing identity management*, Information Security Technical Report **9** (2004), 35–44.
- [Hed12] Ali Hedayati, *An analysis of identity theft: Motives, related frauds, techniques and prevention*, Journal of Law and Conflict Resolution **4** (2012), 1–12.
- [Hil06] Mireille Hildebrandt, *Profiling: From data to knowledge – The challenges of a crucial technology*, Datenschutz und Datensicherheit **30** (2006), no. 9, 548–552.
- [Hog09] Giles Hogben, *Security issues in the future of social networking*, W3C Workshop on the Future of Social Networking, 2009.
- [HP12] John L. Hennessy and David A. Patterson, *Computer architecture – A quantitative approach (5. ed.)*, Morgan Kaufmann, 2012.
- [HR04] Michael Huth and Mark Ryan, *Logic in computer science: Modelling and reasoning about systems*, Cambridge University Press, 2004.
- [HS06] Giles Hogben and Dieter Sommer, *A meta-data and reasoning framework for open assertion and evidence exchange and query*, Technical Report RZ3674, IBM Research, Zurich Research Laboratory (report jointly done with Joint Research Centre), October 2006, <http://domino.research.ibm.com/library/cyberdig.nsf/80741a79b3d5f4d085256b3600733b05/b393d1ec2360ee9c8525722c005b7428!OpenDocument>.
- [Int09] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), *ISO-compliant driving licence, Parts 1–3*, 2005–2009, ISO/IEC Joint Technology Committee 17, Working Group 10.
- [ISO04] ISO, *ISO 8601:2004 Data elements and interchange formats – Information interchange – Representation of dates and times*, ISO Standard 8601:2004, 2004.
- [ISO11] ISO/IEC, *Information technology – Security techniques – Anonymous entity authentication*, ISO/IEC NP 20009 by ISO/IEC JTC 1, 2011.
- [ISO13] ———, *Information technology – Security techniques – Anonymous digital signatures*, ISO/IEC DIS 20008 by ISO/IEC JTC 1, 2013.
- [ITU02] ITU Radiocommunication Assembly, *Standard-frequency and time-signal emissions*, Recommendation ITU-R TF.460-6, 2002.
- [JWG06] Ari Juels, Marianne Winslett, and Atsuhiko Goto (eds.), *Proceedings of the 2006 Workshop on Digital Identity Management, Alexandria, VA, USA, November 3, 2006*, ACM, 2006.

- [Kal97] Burton S. Kaliski Jr. (ed.), *Advances in cryptology - crypto '97, 17th annual international cryptology conference, santa barbara, california, usa, august 17-21, 1997, proceedings*, Lecture Notes in Computer Science, vol. 1294, Springer, 1997.
- [KN03a] Chris Kaler and Anthony Nadalin, *Web Services Federation Language (WS-Federation)*, July 2003.
- [KN03b] ———, *WS-Federation Active Requestor Profile, version 1*, July 2003.
- [Koh10] Markulf Kohlweiss, *Cryptographic protocols for privacy enhanced identity management*, Phd thesis, KU Leuven, March 2010.
- [KTV10] Ralf Küsters, Tomasz Truderung, and Andreas Vogt, *Accountability: definition and relationship to verifiability*, ACM Conference on Computer and Communications Security (Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, eds.), ACM, 2010, pp. 526–535.
- [LGF00] Ninghui Li, Benjamin N. Grosf, and Joan Feigenbaum, *A practically implementable and tractable delegation logic*, IEEE Symposium on Security and Privacy, IEEE Computer Society, 2000, pp. 27–42.
- [LGF03] Ninghui Li, Benjamin N. Grosf, and Joan Feigenbaum, *Delegation logic: A logic-based approach to distributed authorization*, ACM Transactions on Information and System Security (TISSEC) **6** (2003), no. 1, 128–171.
- [LL94] Chae Hoon Lim and Pil Joong Lee, *More flexible exponentiation with precomputation*, in Desmedt [Des94], pp. 95–107.
- [LLW05] Jiangtao Li, Ninghui Li, and William H. Winsborough, *Automated trust negotiation using cryptographic credentials*, ACM Conference on Computer and Communications Security (Vijay Atluri, Catherine Meadows, and Ari Juels, eds.), ACM, 2005, pp. 46–57.
- [LLW09] ———, *Automated trust negotiation using cryptographic credentials*, ACM Trans. Inf. Syst. Secur. **13** (2009), no. 1.
- [LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf, *Pseudonym systems*, Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography (London, UK), SAC '99, Springer-Verlag, 2000, pp. 184–199.
- [MA10] Naveen Mishra and April Adams, *User survey analysis: Ten things you need to know if you sell into the data center*, Gartner Report G00208341, 27 October 2010.
- [Mau10] Ueli Maurer, *Constructive cryptography – A primer*, Financial Cryptography and Data Security (Radu Sion, ed.), Lecture Notes in Computer Science, vol. 6052, Springer Berlin Heidelberg, 2010, pp. 1–1.

- [Mau11] Ueli Maurer, *Constructive cryptography – A new paradigm for security definitions and proofs*, TOSCA (Sebastian Mödersheim and Catuscia Palamidessi, eds.), Lecture Notes in Computer Science, vol. 6993, Springer, 2011, pp. 33–56.
- [McC10] William McCune, *Prover9 and Mace4*, <http://www.cs.unm.edu/~mccune/prover9/>, 2005–2010.
- [ME04] Frank Manola and Eric Miller (Eds.), *Resource Description Framework (RDF)*, World Wide Web Consortium (W3C), W3C Recommendation, <http://www.w3.org/TR/rdf-primer/>, 10 February 2004.
- [Men87] Elliott Mendelson, *Introduction to mathematical logic (3. ed.)*, Chapman and Hall, 1987.
- [MRT12] Ueli Maurer, Andreas Rüedlinger, and Björn Tackmann, *Confidentiality and integrity: A constructive perspective*, TCC (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 7194, Springer, 2012, pp. 209–229.
- [MS94] Ueli M. Maurer and Pierre E. Schmid, *A calculus for secure channel establishment in open networks*, ESORICS (Dieter Gollmann, ed.), Lecture Notes in Computer Science, vol. 875, Springer, 1994, pp. 175–192.
- [MS96] ———, *A calculus for security bootstrapping in distributed systems*, Journal of Computer Security **4** (1996), no. 1, 55–80.
- [MS10a] Samuel Muller and Dieter M. Sommer, *Automated relationship management for electronic social networks*, United States Patent Application Publication, No. US 2010/0235886 A1, September 16, 2010.
- [MS10b] ———, *Context-aware electronic social networking*, United States Patent Application Publication, No. US 2010/0228590 A1, September 9, 2010.
- [MS12a] ———, *Access to electronic social networks*, United States Patent, No. US 8,234,300 B2, July 31, 2012.
- [MS12b] ———, *Negotiable information access in electronic social networks*, United States Patent Application Publication, US 2012/0297003 A1, November 22, 2012.
- [MV09a] Sebastian Mödersheim and Luca Viganò, *The open-source fixed-point model checker for symbolic analysis of security protocols*, FOSAD (Alessandro Aldini, Gilles Barthe, and Roberto Gorrieri, eds.), Lecture Notes in Computer Science, vol. 5705, Springer, 2009, pp. 166–194.

- [MV09b] Sebastian Mödersheim and Luca Viganò, *Secure pseudonymous channels*, Proceedings of the 14th European conference on Research in computer security (Berlin, Heidelberg), ESORICS'09, Springer-Verlag, 2009, pp. 337–354.
- [MvHE04] Deborah L. McGuinness and Frank van Harmelen (Eds.), *OWL Web Ontology Language*, World Wide Web Consortium (W3C), W3C Recommendation, <http://www.w3.org/TR/owl-features/>, 10 February, 2004.
- [MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press, August 2001.
- [Nat09] National Institute of Standards and Technology (NIST), *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186–3, June 2009.
- [Nat10] ———, *NSTIC: National Strategy for Trusted Identities in Cyberspace*, <http://www.nist.gov/nstic/>, 2010.
- [OEC80] OECD, *OECD guidelines on the protection of privacy and transborder flows of personal data*, OECD guidelines, September 1980.
- [OEC08] OECD, *OECD policy guidance on online identity theft*, 2008.
- [Ohm10] Paul Ohm, *Broken promises of privacy: Responding to the surprising failure of anonymization*, UCLA Law Review **57** (2010), 1701.
- [Osa11] Juro Osawa, *As Sony counts hacking costs, analysts see billion-Dollar repair bill*, The Wall Street Journal, May 9, 2011.
- [Ped91] Torben P. Pedersen, *Non-interactive and information-theoretic secure verifiable secret sharing*, CRYPTO (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 576, Springer, 1991, pp. 129–140.
- [Pfe04] Frank Pfenning, *Automated theorem proving*, Lecture Notes, Carnegie Mellon University, USA, 2004.
- [PH] Andreas Pfitzmann and Marit Hansen, *A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management*.
- [PH10] ———, *A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management*, Tech. report, Technische Universität Dresden, August 2010, v0.34.
- [Pip76] Nicholas Pippenger, *On the evaluation of powers and related problems (preliminary version)*, Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1976), October 1976.

- [PKK⁺11] Bart Priem, Eleni Kosta, Aleksandra Kuczerawy, Jos Dumortier, and Ronald Leenes, Digital privacy (Jan Camenisch, Dieter Sommer, and Ronald Leenes, eds.), Springer-Verlag, Berlin, Heidelberg, 2011, pp. 91–106.
- [PLFK11] Bart Priem, Ronald Leenes, Alea Fairchild, and Eleni Kosta, *The need for privacy-enhancing identity management*, Digital Privacy (Jan Camenisch, Ronald Leenes, and Dieter Sommer, eds.), Lecture Notes in Computer Science, vol. 6545, Springer Berlin Heidelberg, 2011, pp. 53–71.
- [PRI08] PRIME Consortium, *PRIME project Web site*, 2008, www.prime-project.eu.
- [Pri10] PrimeLife Consortium, *PrimeLife project Web site*, 2010, www.primelife.eu.
- [Rep12] Republic of Singapore, *Personal Data Protection Act 2012*, Government Gazette – Acts Supplement (December 7, 2012).
- [RKW12] Franziska Roesner, Tadayoshi Kohno, and David Wetherall, *Detecting and defending against third-party tracking on the web*, Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (Berkeley, CA, USA), NSDI '12, USENIX Association, 2012, pp. 155–168.
- [Rob65] John Alan Robinson, *A machine-oriented logic based on the resolution principle*, Journal of the ACM **12** (1965), 2341.
- [RS09] William Roberds and Stacey L. Schreft, *Data breaches and identity theft*, The Eighth Workshop on the Economics of Information Security, 2009.
- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.
- [RSA83] ———, *A method for obtaining digital signatures and public-key cryptosystems (reprint)*, Communications of the ACM **26** (1983), no. 1, 96–99.
- [RSA10] Sasha Romanosky, Richard Sharp, and Alessandro Acquisti, *Data breaches and identity theft: When is mandatory disclosure optimal?*, WEIS, 2010.
- [RW69] George A. Robinson and Lawrence Wos, *Paramodulation and theorem-proving in first-order theories with equality*, Machine Intelligence **4** (1969), 135–150.

- [Sch] Stephan Schulz, *The E Theorem Prover*, <http://eprover.org/> or <http://www4.informatik.tu-muenchen.de/~schulz/E/E.html>.
- [Sch91] Claus-Peter Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology **4** (1991), no. 3, 161–174.
- [Sch02] Stephan Schulz, *E – A brainiac theorem prover*, AI Communications **15** (2002), no. 2-3, 111–126.
- [Sch09] Christoph Schnabel, *Privacy and data protection in EC telecommunications law*, EC Competition and Telecommunications Law, Kluwer Law International BV, 2009, pp. 509–568.
- [Sec10] Security Team, IBM Research Zurich, *Specification of the Identity Mixer cryptographic library*, Research Report RZ 3730, IBM Research Division, IBM Research Zurich, April 2010, IBM Research Report RZ 3730.
- [Sha79] Adi Shamir, *How to share a secret*, Communications of the ACM **22** (1979), no. 11, 612–613.
- [SM09] Dieter M. Sommer and Samuel Muller, *Hybrid profile management for electronic social networks*, United States Patent Application Publication, No. US 2009/0248844 A1, October 1, 2009.
- [Sol07] Daniel J. Solove, *‘I’ve got nothing to hide’ and other misunderstandings of privacy*, San Diego Law Review **44** (2007), 745.
- [Som04a] Dieter Sommer, *Efficient exponentiation and multi-exponentiation in finite groups*, Master’s thesis, Universität Klagenfurt, 2004.
- [Som04b] ———, *PRIME Architecture V0*, PRIME project, 2004.
- [Som05] ———, *PRIME Architecture V1*, PRIME project, 2005.
- [Som07] ———, *Identity Mixer library implementation*, Presented at Internet Identity Workshop 2007 (IIW 2007), Maintain View, CA, USA, as part of an identity management prototype, 2007, IBM-internal cryptographic code.
- [Som08] ———, *PRIME Architecture V3*, PRIME project, 2008.
- [Som11] ———, *Architecture*, Digital Privacy (Jan Camenisch, Ronald Leenes, and Dieter Sommer, eds.), vol. 6545, Springer Berlin Heidelberg, 2011, pp. 151–288.
- [SPA10] SPASS Team, *SPASS: An automated theorem prover for first-order logic with equality*, Web site: <http://www.spass-prover.org/>, 2010.

- [SW11] Ian Sherr and Nick Wingfield, *Play by play: Sony's struggles on breach*, The Wall Street Journal, <http://online.wsj.com/article/SB10001424052748704810504576307322759299038.html>, May 7, 2011.
- [SYT09] Geoff Sutcliffe, Aparna Yerikalapudi, and Steven Trac, *Multiple answer extraction for question answering with automated theorem proving systems*, FLAIRS Conference (H. Chad Lane and Hans W. Guesgen, eds.), AAAI Press, 2009.
- [The06] The European Parliament and the Council of the European Union, *Directive 2006/24/EC of the European Parliament and of the Council of 15 March 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC*, Official Journal L **54** (2006).
- [The11] The White House, *National strategy for trusted identities in Cyberspace – enhancing online choice, efficiency, security, and privacy*, Report of The White House, http://www.whitehouse.gov/sites/default/files/rss_viewer/NSTICstrategy_041511.pdf, 2011.
- [The12] ———, *Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy*, February 2012.
- [Tra12] Tracking Protection Working Group, *Do Not Track*, World Wide Web Consortium (W3C), <http://www.w3.org/2011/tracking-protection/>, 2012.
- [VD10] Jennifer Valentino-Devries, *How to avoid the prying eyes*, Article series in The Wall Street Journal, <http://online.wsj.com/article/SB10001424052748703467304575383203092034876.html>, 2010.
- [vdBL10] Bibi van den Berg and Ronald Leenes, *Audience segregation in social network sites*, SocialCom/PASSAT (Ahmed K. Elmagarmid and Divyakant Agrawal, eds.), IEEE Computer Society, 2010, pp. 1111–1116.
- [vdBL11] ———, *Keeping up appearances: Audience segregation in social network sites*, Computers, Privacy and Data Protection (Serge Gutwirth, Yves Poulet, Paul De Hert, and Ronald Leenes, eds.), Springer, 2011, pp. 211–231.
- [Ver12] Verizon, *2012 data breach investigations report*, http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2012-ebk_en_xg.pdf, 2012.

- [Ver13] ———, *2013 data breach investigations report*, <http://www.verizonenterprise.com/DBIR/2013/>, 2013.
- [VPCS11a] Chris Vanden Berghe, Tadeusz J. Pietraszek, Jan L. Camenisch, and Dieter Sommer, *Privacy enhanced comparison of data sets*, United States Patent, No. US 7,974,406 B2, Jly 5, 2011.
- [VPCS11b] ———, *Privacy enhanced comparison of data sets*, United States Patent, No. US 7,974,407 B2, July 5, 2011.
- [WAFH11] Erik Wästlund, Julio Angulo, and Simone Fischer-Hübner, *Evoking comprehensive mental models of anonymous credentials*, iNetSec (Jan Camenisch and Dogan Kesdogan, eds.), Lecture Notes in Computer Science, vol. 7039, Springer, 2011, pp. 1–14.
- [Wal11] Michał Walicki, *Introduction to mathematical logic*, World Scientific Publisher, December 2011.
- [WB90] Samuel D. Warren and Louis D. Brandeis, *The right to privacy*, Harvard Law Review 4 (Dec 15, 1890).
- [WDF⁺09] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischniewski, *SPASS Version 3.5*, Automated Deduction – CADE-22 (Renate A. Schmidt, ed.), Lecture Notes in Computer Science, vol. 5663, Springer Berlin Heidelberg, 2009, pp. 140–145.
- [Wes67] Alan F. Westin, *Privacy and freedom*, Atheneum, 1967.
- [WL02] William H. Winsborough and Ninghui Li, *Towards practical automated trust negotiation*, POLICY, IEEE Computer Society, 2002, pp. 92–103.
- [WL04] ———, *Safety in automated trust negotiation*, IEEE Symposium on Security and Privacy, IEEE Computer Society, 2004, pp. 147–160.
- [WR73] Lawrence Wos and George A. Robinson, *Maximal models and refutation completeness: Semidecision procedures in automatic theorem proving*, Word Problems: Decision Problems and the Burnside Problem in Group Theory, W. W. Boone, F. B. Cannonito, and R. C. Lyndon, eds., North-Holland, Amsterdam, 1973, pp. 609–639.
- [WSJ00] William H. Winsborough, Kent E. Seamons, and Christine E. Jones, *Automated trust negotiation*, Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), vol. 1, 2000, pp. 88–102.
- [WSS04] *Web Services Security (WS-Security)*, OASIS Standard, 2004.
- [Zim95] Philip R. Zimmermann, *The official PGP users guide*, MIT Press, Cambridge, MA, USA, 1995.

- [Zur11] Zurich American Insurance Corporation, *Data breach cost – Part one: risks, costs and mitigation strategies for data braches*, Report, 2011.

Curriculum Vitae

The author's CV has been removed for the purpose of data protection as recommended for the electronic publication of dissertations at TU Darmstadt.



TECHNISCHE
UNIVERSITÄT
DARMSTADT

© June 2014